Tampereen yliopisto

Julius Tamminen

EVALUATING THE PERFORMANCE OF SECURE GATEWAY BETWEEN REAL-TIME SIMULATION NETWORKS

Master's Thesis
Faculty of Information Technology and Communication Sciences
Examiners: Prof. Mikko Valkama
Dr. Joonas Säe
September 2021

ABSTRACT

Julius Tamminen: Evaluating the performance of secure gateway between real-time simulation networks

Master's Thesis
Tampere University
Master's Degree Program in Information Technology
September 2021

The current trend of digitalization and automation continues strongly onward within all industries, which creates innovations and improvements to existing solutions. In the field of communication and networking, there are many new and old network solutions, that each can be utilized in various ways and have their own unique purposes and configurations. One network can connect and serve multiple users allowing communication between them, whereas some networks can simulate real-life scenarios and perform complicated calculations. In addition, these networks can co-operate and share mutual information, if they are connected with proper equipment such as gateways. These connections allow the creation of even wider networks, which can enable use cases, where multiple organizations can share data and operate together. One example of such massive scale networks is the Live Virtual Constructive (LVC) concept, where real time live and simulated events are combined to the same environment, which can be accessed from different geographical locations.

However, when connecting two or more networks together, the information security aspect must be considered, especially when some of the networks contains more sensitive data than the other networks. The security aspect can be fulfilled, and the leak of the sensitive data to other networks can be prevented by utilizing a secure gateway that performs the necessary filtering operations. Nevertheless, the secure gateway must consider the performance requirements of the networks, which can vary depending on the end purpose and the use case of the networks. The performance requirements, in the context of real-time simulation networks, were determined by the latency and the throughput benchmarks that the real-time simulation network must fulfill. The benchmark values were established by analyzing the literature and studies, with the help of a commercial simulation system and by interviewing the experts of the field.

Once the performance requirement benchmarks were found out, the final evaluation of the effect of a real-world secure gateway implementation, Cross Domain Solution (CDS) made by Insta DefSec, could be made. The results for the evaluation were made by executing the latency and throughput tests in separate measurement set-ups: one measurement set-up measured the latency, and the other set-up measured the maximum throughput of the network, when the CDS was attached and connected between the two real-time simulation networks. Then, the same measurement set-ups were repeated by replacing the CDS with a commercial protocol translation and bridging software VR-Exchange, which only passed the data through it, to determine the reference results.

For the performance requirements it was found out, that the throughput requirement would be 1870 entities/s, whereas the upper limit for the latency would be 100 ms. The throughput measurement disclosed that the maximum throughput of the CDS was 386.4 entities/s, while the VR-Exchange resulted in 1077 entities/s. Furthermore, the latency measurement with a single updating entity resulted in an average latency of 3.9 ms for the CDS and 13.7 ms for the reference. Thus, the performance requirements were achieved only partially, as the throughput requirement was not reached. Nevertheless, the latency of the CDS was below the requirement of 100 ms, when a single updating entity was measured. Thus, the CDS fulfills the latency requirement when the number of simultaneously updating entities is minimal. However, optimization of the software of the CDS would improve the throughput and the latency capabilities even further.

Keywords: secure gateway, real-time simulation networks, latency, throughput, HLA

TIIVISTELMÄ

Julius Tamminen: Reaaliaikaisten simulaatioverkkojen välisen tietoturvallisen yhdyskäytäväratkaisun suorituskyvyn arvioiminen

Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Syyskuu 2021

Digitalisaation ja automatisoinnin aikakausi jatkuu yhä vahvana kaikilla teollisuuden aloilla, mikä luo yhä uusia innovaatioita ja parannuksia jo olemassa oleviin ratkaisuihin. Tietoliikennetekniikan ja -verkkojen alalla on paljon uusia ja vanhoja verkkototeutuksia, joita voidaan käyttää eri tavoin ainutlaatuisissa käyttötarkoituksissa ja konfiguraatioissa. Tietoverkkoja voidaan käyttää esimerkiksi yhdistämään eri toimijoita, mikä mahdollistaa kommunikoinnin niiden välillä. Toisaalta tietoverkkoa voidaan käyttää myös simulointiin, mallintamiseen ja laskentaan. Lisäksi näitä toimintoja ja tietoverkkoja voidaan käyttää myös yhdessä, jos tietoverkot on yhdistetty oikein esimerkiksi yhdyskäytävien avulla. Yhdistelemällä eri tietoverkkoja voidaan mahdollistaa yhä laajempien verkostojen hyödyntämisen eri käyttötarkoituksissa, joissa esimerkiksi eri organisaatiot jakavat tietoa ja tekevät yhteistyötä. Yksi tämän kokoluokan käyttötapaus on LVC-konsepti, jossa yhdistetään reaaliaikainen toiminta simuloitujen ja elävien toimijoiden välillä samaan ympäristöön, johon voidaan ottaa myös yhteys maantieteellisesti eri sijainneista.

Kahta tai useampaa eri tietoverkkoa yhdistäessä on kuitenkin huomioitava myös tietoturvanäkökulma, etenkin jos jossain tietoverkossa käsitellään luottamuksellisempaa tietoa kuin toisissa. Käyttämällä tietoturvallisia yhdyskäytäviä, jotka osaavat suodattaa tiettyjä sanomia tai informaatiota, tietovuodot voidaan estää ja tietoturvanäkökulma voidaan ottaa huomioon. Tietoturvalliset yhdyskäytävät eivät kuitenkaan saa rajoittaa tietoverkon suorituskykyä liikaa, ja eri tietoverkkojen suorituskykyvaatimukset tulee huomioida tapauskohtaisesti. Suorituskykyvaatimusten tulisi täyttää tietyt latenssi- ja läpäisykykyvaatimukset, jotta reaaliaikaista tietoa pystyttäisiin hyödyntämään tehokkaasti. Näille latenssi- ja läpäisykykyvaatimuksille määriteltiin raja-arvot kirjallisuuden, tutkimusten, kaupallisen simulaattorijärjestelmän ja asiantuntijoiden haastatteluiden avulla.

Suorituskykyvaatimusten määrittelyn jälkeen Insta DefSec:in luomaa tietoturvallista yhdyskäytäväratkaisua, Cross Domain Solution:ia (CDS), ja sen vaikutusta tietoverkon suorituskykyyn pystyttiin arvioimaan latenssi- ja läpäisykykymittausten avulla. Tietoverkon latenssia ja läpäisykykyä mitattiin eri skenaarioissa, joissa CDS yhdistää kaksi eri tietoverkkoa. Lisäksi suoritettiin referenssimittaukset, jossa CDS korvattiin kaupallisella protokollien käännös- ja välitysohjelmalla VR-Exchangella, joka päästi tiedon lävitseen ilman suodatuksia tai muita muokkausoperaatioita.

Suorituskykyvaatimuksiksi muodostui lopulta läpäisykykyvaatimus 1870 entiteettiä/s, ja latenssivaatimus 100 ms. Läpäisykykymittauksista selvisi, että CDS:n läpäisykyvyn maksimiarvo oli 386.4 entiteettiä/s, ja VR-Exchangen vastaava arvo oli 1077 entiteettiä/s. Lisäksi latenssimittaukset tuottivat CDS:n latenssin keskiarvoksi 3.9 ms ja referenssille 13.6 ms. Täten tulosten perusteella suorituskykyvaatimukset täyttyivät vain osittain, sillä läpäisykykyvaatimusta ei saavutettu. CDS:n latenssi oli kuitenkin pienempi kuin 100 ms, kun mitattiin yhtä päivittyvää entiteettiä. Tällöin CDS saavuttaa latenssivaatimuksen, kun samanaikaisesti päivittyvien entiteettien määrä on minimaalinen. Kehittämällä ja optimoimalla CDS:n ohjelmistoa voitaisiin parantaa sen läpäisykyvyn ja latenssin suorituskykyä entistä paremmaksi.

Avainsanat: Tietoturvallinen yhdyskäytävä, reaaliaikainen simulaatioverkko, latenssi, läpäisykyky, HLA

PREFACE

I would honorably thank all the people, who supported me and helped me to complete this thesis within its various phases. Especially, I would express my gratitude to Marko Latvala, who guided me with his professional knowledge, and kept me on the right track when implementing the measurement system and algorithms, and while forming the performance requirements. In addition, the excellent help and mentoring from Joonas Säe enabled the academic accuracy of this thesis. Joonas and the staff from the former department of Electrical and Communications Engineering made my study and working periods enjoyable and memorable in the Tampere University over the years. I might be back for the postgraduate studies later, but the future remains still open for everything. Furthermore, also the colleagues at Insta DefSec have helped me to solve various problems during this thesis, and without them the integration and debugging of the measurement software would still continue at the moment. Finally, I am grateful for the love and support that Kaisla and my family have given to me during the time of this project.

In Tampere, 24.09.2021

Julius Tamminen

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

3GPP	3rd Generation Partnership Project
ALSP	Aggregate Level Simulation Protocol
API	Application Programming Interface
ASTERIX	All Purpose Structured Eurocontrol Surveillance Information Exchange
CDF	Cumulative distribution function
CDS	Cross Domain Solution
CSV	Comma-Separated Values
DDM	Data distribution management
DIS	Distributed Interactive Simulation
DM	Declaration management
DMSO	Defense Modeling and Simulation Office
FEDEP	Federation Development and Execution Process
FOM	Federation Object Model
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
IEEE SA	IEEE Standards Association
ISO	International Organization for Standardization
LVC	Live Virtual Constructive
M&S	Modeling and Simulation
OMT	Object Model Template
OOP	Object-Oriented Programming
PAR	Project Authorization Request
RevCom	Standards Review Committee
RO	Receive-Order
RPR FOM	Real-time Platform Reference Federation Object Model
RTI	Run Time Infrastructure
RTT	Round Trip Time
SISO	Simulation Interoperability Standards Organization
SOM	Simulation Object Model
TSO	Time-Stamp-Order
XML	Extensible Markup Language

1. INTRODUCTION

In recent years, digitalization has been growing and the usage of automated and digital applications has increased within industry, but also in other fields such as entertainment and education. One of the major beneficiaries of this trend is the Modeling and Simulation (M&S) industry which focuses on recreating different real-life events in digital environments with the help of mathematical models [1]. These independent simulations can adapt very complex calculations and perform intensive computing, which can yield lifelike experiences in the form of vehicle simulators and virtual reality worlds for example.

However, it is not a trivial task to implement a proper simulation that can model the events well. The task becomes even more challenging, when two or more simulations that are dissimilar, must communicate with each other. Common time synchronization, the optimization of calculations and the connections for allowing the interoperability are one of the key challenges when combining two or more simulation networks together [2]. Fortunately, there are already solutions to combine the different networks and avoid these challenges. By using standardized protocols, such as High Level Architecture (HLA), the communication between individual simulation systems is possible, even if their main purposes differ significantly. Nevertheless, when combining two or more different networks together, the information security aspect must be considered. The combination of two or more networks allows sharing of all the data between them, which is not preferable, if some information is confidential or should be used only in one of the networks.

To avoid this issue, secure gateway solutions can be used as the combining element between the simulation networks. Insta DefSec's Cross Domain Solution (CDS) is a secure gateway, that enables the filtering or modification of the confidential data between the networks it is connected to. This ability makes possible the secure connection between two or more networks with different security classifications. Thus, the sensitive part or the whole message can be filtered, but at the same time, other messages can pass the gateway as they are without any altering.

Besides the information security point of view, also the performance requirements of the simulation networks must be taken into account. Especially real-time simulation systems

require relatively small delays for the exchange of data, to be able to simulate the scenarios and properties correctly. Another key performance requirement for real-time simulation systems is high enough capacity or throughput capability to handle the transfer of numerous messages, entity updates, various computations and other tasks during the process of simulation.

One of the main goals of this thesis is to find out appropriate scale or tolerable limits for the performance requirements of the real-time simulator networks. Suitable references are searched with the help of the existing literature, standards, existing commercial real-time simulation systems and the experts of the industry. When the base for the performance requirements is found, the next task is to evaluate whether the real-time simulation network with a CDS between is capable to achieve these results. The performance of the real-time simulation network should not drop below the tolerable bounds, when CDS is applied to the system.

The test measurements and set-ups for evaluating the effect of adding the CDS to the real-time simulation network are performed in the empirical part of this thesis. After all, the performance requirements and the measurements together would disclose, whether the usage of the CDS between the simulation networks would be possible in operational use cases requiring data with real life and real-time constraints. Alternatively, if the performance requirements are not met, valuable information for the further development of the CDS is gathered to enable more precise improvements for its current implementation.

2. SIMULATION SOFTWARE

The digitalization has increased the demand to experience various real-life events and activities virtually. Simulation software try to imitate these events and activities in such ways that the environment and events would be as realistic as the real experience would be. To understand the concept of simulation, especially in the context of this thesis, we address next the definition of simulation and its advantages, the process of creating new simulation standards, and the details about the main simulation standard behind the final measurement process.

2.1 Simulations and their advantages

The term simulation can be defined as the imitation or copying of a real-world process, an operation or a system over some time. The simulation requires an artificial timeline, where each event can be put, to observe and analyze the causality between them. [3] Simulations can be used in different environments for various purposes: educational simulations such as driving simulators can be helpful in the process of learning to drive. If the same driving simulator is integrated to a video game, the context of the simulation changes to entertainment purposes. Furthermore, simulations are utilized in the field of research and development, where existing systems and conceptual models are simulated with the help of computers and software, like MATLAB and SolidWorks.

There are several advantages that have helped the simulations to become as widely used in different contexts as they are nowadays. One of the biggest advantages is the possibility to choose different outcomes, without needing to allocate any real resources to them [3]. This means in practice that different options can be tested beforehand and the best alternative is chosen from the simulated resources. For example, choosing the best material for a building could be very expensive and time consuming, if the different options are tested and built in real-life, compared to simulating the different alternatives. Another advantage is the ability to stop, speed up, or slow the time in the simulation [3]. Noteworthy events can be revisited and the investigation of why they happened becomes easier, whereas the unnecessary ones can be skipped. However, the time management and manipulation procedures are not so simple if the data of the simulator is needed and handled real-time.

In addition, the training of different operators can be done in the simulators, which reduces the overall expenses of the training [3]. For example, considering the training of

the aircraft pilots, fuel consumption is not an expense anymore. Also, some situations including safety aspects and experimental maneuvers are more feasible to do with simulations, like practicing the emergency landings in various scenarios, without any risks for damaging the aircraft or harming the pilot. [4] On the other hand, the developing process of simulations can be time consuming, expensive and it may require specific knowledge and expertise from the field. Also, the results from the simulations can be difficult to interpret, due to complexity of the system and the amount of information the simulation can produce. [3][4] All in all, there are more advantages than disadvantages, which make the usage of simulators feasible in many fields, especially as the technologies evolve and develop forward.

2.2 Standards and the process of standardization

There are many contributors and manufacturers in the field of simulators, but also in the field of communication and industry in general. In order to guarantee that their components and software are compatible with each other, they must follow commonly approved principles called standards. These standards are defined by the standardization organizations, such as Institute of Electrical and Electronics Engineers (IEEE), 3rd Generation Partnership Project (3GPP) and International Organization for Standardization (ISO), which are considerably massive organizations. Next, we address the IEEE's process of standardization, as the development of the key standard of this thesis, the HLA, is controlled by this standardization organization.

The standardization process of IEEE consists of six parts: the initiation of the project, mobilizing the working group, drafting the standard, balloting the standard, gaining final approval, and maintaining the standard as presented in the Figure 1 [5].

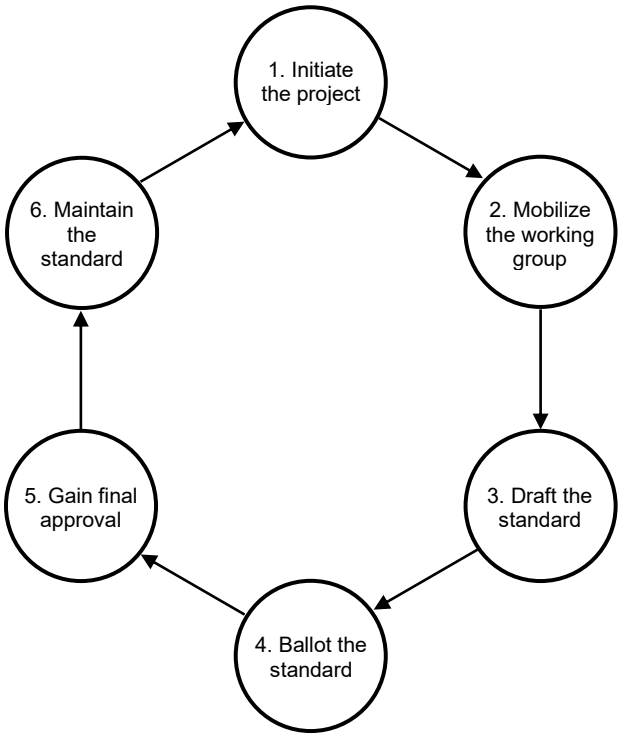

Figure 1: *The standardization process of IEEE. [5]*

The process of standardization starts from the initiation of the project. Possibly a new idea or implementation that develops the current models or completely new state-of-the-art solution can lead to submitting a Project Authorization Request (PAR). Mainly the reason and goals of the standardization project are stated in the PAR which is a legal document that has to be submitted to IEEE Standards Association (IEEE SA) for the first approval. After the approval, the process moves to the second part, mobilizing the working group. The working group are the people who will write the standard and manage its progress. The working group needs at least one working group Chair that leads the process towards the next stages. In addition to the Chair, there are the individuals who participate in the writing and decision making of the standard. These individuals can be anyone having enough expertise in the field of the requested standard. If the working group becomes large and the managing gets difficult, the working group can have additional officers helping to share the load. The Vice Chair can assist the Chair's tasks, the Treasurer can handle the fees of the meetings and other expenses, and the Secretary can take the notes and other records of the meetings and forward them onwards.

When the working group has been mobilized, the drafting of the standard can commence. After drafting and having the writing process in the finalization phase, the working group will create the balloting group, that examines, comments and then votes for the approve or the disapprove of the standard. If consensus is achieved by having a

minimum of 75 % of votes of the ballots to approve, the balloting is approved. If not, the disapprovals have requested changes with comments, which must be handled. After the new changes, the balloting process is recirculated, and the consensus is revisited.

When the technical part of the standard has been approved in the balloting process, the other aspects, such as documentation, IEEE SA procedures and openness are reviewed by the Standards Review Committee (RevCom). If there are not any insufficiencies, the RevCom will recommend this standard for the IEEE SA and the standard gets the final approval. However, the standardization process will not end here, as it must be maintained. The standard has a validity period of 10 years after the final approval. After the validity, the standard must be revised or withdrawn. Therefore, in order to keep the standard up to date, the work for the new revised version of the standard must begin, which forms the cyclic and recurring process of standardization.

2.3 High Level Architecture (HLA)

High Level Architecture (HLA) is one standardized solution to connect different simulator systems together via common integrated architecture and interface [6]. The first stable version of HLA was 1.0, which was developed by the Defense Modeling and Simulation Office (DMSO) in the 1996 [7]. The main motivation for a new simulation protocol was to improve the integration and the co-operation of the military's wide-ranging simulation systems. The development team of HLA used the already existing simulation protocols, Distributed Interactive Simulation (DIS) and Aggregate Level Simulation Protocol (ALSP), to produce a more generic and versatile simulation protocol. [8]

The core structure of the protocol was presented in the first revision containing the framework and the rules, interface specifications and object model specifications of the implementation. After a few years, the next updates were the improved version HLA 1.3 and two Run Time Infrastructure (RTI) solutions RTI 1.3 and RTI 1.3 Next Generation (NG). The next important leap forward for the HLA was when IEEE standardized HLA as *HLA std 1516-2000* to open the usage of HLA to wider audiences, outside of the military context. [7] The standardization also brought up some updates that improved the usage of Federation Object Models (FOMs) with the support of Extensible Markup Language (XML) files. Also, the Federation Development and Execution Process (FEDEP) became clearer with the addition of IEEE 1516.3-2003 to the standard.

The newest revision of HLA is the *HLA 1516-2010*, also known as HLA Evolved. This version came with a major update to the FOM structure enabling modular FOMs meaning that the information that is exchanged within the simulation can be more flexible and

optimized for each separate simulator system [9]. Currently, the next revision of HLA is in development with the working group lead by the Simulation Interoperability Standards Organization (SISO). One of the upcoming new updates is adding authorization mechanisms to determine which entities have access to join to the integrated system of simulators. [10]

2.3.1 Structure

The overall functioning HLA simulation consist of federates, federations and the Run Time Infrastructure (RTI). The interoperation of all these entities together over time form the federation execution process. The RTI is the heart of the federation execution because it acts as a central unit, which manages data exchange, joining or resigning and other key services between the federates that it connects. All the services that the RTI provides and how the federates must communicate with it can be found from the HLA interface specification which is addressed later in the chapter 2.3.3.

A federate is an application, possibly a simulator, that wants to communicate with other federates to accomplish its tasks as defined. [6] The other federates can be within the same federation or in the other federations. If the federates are within the same federation, they must comply with the same federation wide FOM, which defines all the possible object classes, instance attributes, data types and interaction parameters to be used. Even though the HLA does not take a stand on what type of data the FOM can contain, it defines the overall structure of the FOM. This structure for presenting the object classes, instance attributes, data types and interaction parameters is specified in the Object Model Template (OMT), which is covered in the chapter 2.3.4. Thus, all the federates shall follow the OMT principles regardless of their federation. However, if the federates are within the same federation, it usually means that the federates have similar applications or are the same application’s different instances or positions.

Similar federates can have also differing rulesets due to the federate wide Simulation Object Models (SOMs). The SOMs are like FOMs, but they define the object classes, instance attributes, data types and interaction parameters for the individual federate [6]. For example, the two similar federates within the same federation can have the same object classes, but the other implements the class with different attributes as the other. However, all the attributes must be defined in the FOM of the federation.

Regardless of the federations, the data exchange between the federates goes always through the RTI, with the help of update and reflection messages, which is discussed in detail later. Nevertheless, the main benefit of the HLA is the generalization of the

interfaces: each federate operates with the same interface, even though the federate applications can be very different from each other. This overall generalization enables the interoperability between multiple different simulators, which provides one solution for managing modern complex concepts like Live Virtual Constructive (LVC) where real time live events and simulated ones are combined to the same environment.

2.3.2 Framework and rules

IEEE 1516–2010 is the first standard out of the three 1516 standards, that HLA applications should follow, when implementing the standard. This standard defines the framework and rules of HLA, by introducing a set of 10 rules. Half of the rules apply to the whole federation level and the remaining half are federate-specific. [6]

The first rule: “Federations shall have an HLA FOM, documented in accordance with the HLA OMT” means, that within each separate federation, there must be a common ruleset for all the federates [6]. The ruleset, or how the data is exchanged between the federates and the RTI, is defined in the FOM. The data itself can vary because HLA standard does not define it. However, the structure must comply with the HLA OMT, which defines the general, reusable and inheritable object model structure for HLA. This type of reusability is in the heart of design of the HLA. One example of FOM is the SISO’s Real-time Platform Reference Federation Object Model (RPR FOM), which defines the objects and hierarchy for the simulated physical entities, like vehicles and lifeforms, and interactions including collisions and communications between them [6][11].

Next, the second rule: “In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI” [6]. This statement means, that the RTI is only the forwarder of the data, and the information is stored within the federates and their object’s instances themselves, not in the RTI. Still, the RTI can use information within federates to support its services, but RTI will not do any changes to the data. By having the object and federate-specific information separated from the supporting infrastructure, mainly the RTI, the federation can adapt to very different tasks. The overall flexibility and generalization are the advantages that the HLA utilizes.

The third rule is similar to previous one: “During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI” [6] The communication and data exchange between the federates are managed by the RTI. All the services that RTI provides are listed in the HLA 1516.1-2010 federate interface specification which are discussed in more detail in the next chapter. When a new federate joins to a federation, it communicates with the RTI to get the information it needs and what it provides, to

operate correctly in the federation. Then, the RTI ensures, that the correct data is shared alike with the federates. The data exchange is declared in the FOM, which allows the coherent usage of the data within the federation.

The last two federation-specific rules are also tightly connected with the already presented ones: “During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification” and “During a federation execution, an instance attribute shall be owned by, at most, one joined federate at any given time” [6]. The former specifies, that the communication between the federates and RTI must comply with the standard interface given in the HLA 1516.1-2010 federate interface specification. The main advantage of a common interface is the same as the usage of application programming interfaces (APIs) in programming in general: the interface remains constant, even though there are changes and updates behind the interface in the used application. The latter rule states that, the maximum number of owners to a single instance attribute is one federate, but the owner for the same instance’s other attributes is not required to be the same federate. For example, the location of a vehicle instance can be owned by one federate, whereas the same vehicle’s state of the fuel can be owned by other federate. In addition, the ownerships of the attributes can change dynamically during a federation execution with the help of ownership management services of the RTI.

The remaining rules are federate-specific and therefore applied in the federate level. The sixth rule is as follows: “Federates shall have an HLA SOM, documented in accordance with the HLA OMT” [6]. Similar to rule one, federation having one FOM, each federate should have one SOM. SOM determines the object models and instance attributes that are used within one simulation to storage and distribute information. For instance, the federate models and simulates a vehicle with numerous instance attributes like position, speed, fuel state, assets and usability. Nevertheless, the other federates of the federation are only interested in the usability of the federates vehicle instances. Therefore, only vehicle usability is distributed outside this federate and monitored by the other federates in the federation. To enable this exchange and transfer of data, federation wide common rules must have been made. Thus, setting the vehicle object with its instance attributes to FOM, makes the relevant information exchange possible to all federates needing it.

The following rules, rule 7, 8 and 9 define more requirements for the SOMs of the federates. “Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs”, “Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs” and “Federates shall be able to vary the conditions

(e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs" [6]. As stated in the previous paragraph, that FOM enables the communication between the federates by common rulesets, but additional federate-specific information should be found from the SOM. This information should contain at least, as stated in the rules 7–9, the list of instance attributes that the federate updates or reflects, how often or with what criteria these instance attributes are updated, and the possibility of changing the ownership of the instance attributes to another federates.

The last rule: "Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation" considers the time management of the federates [6]. The federate's local time management system must be implemented in a way that the interoperability between federates can be achieved considering the timestamps and possible inconsistencies with the current time in each federation. Some of the federates may run faster than real time and some does not track the time at all. One solution to synchronize each federate's time to each other is to utilize the time management services of the RTI. The accuracy and synchronization of time play a significant role in the managing of a large real-time simulator system. Even minor delays or asynchronous times can lead to inoperability of the whole federation if the essential information gets delayed, distorted, or lost.

Now, all the 10 basic rules of HLA have been examined and as a summary, they are put together in the Table 1.

Table 1: *The 10 rules of HLA.*

Rule number	Scope	Description
1	Federation	The ruleset for the data and its exchange within each federation is given in the FOM, which follows the restrictions of the OMT.
2	Federation	The information of the objects is stored in the federates, not in the RTI.
3	Federation	The data exchange goes always through the RTI, and cannot happen directly between the federates or the federations.
4	Federation	The communication with the RTI happens through the interface defined by the HLA interface specification.
5	Federation	The maximum number of owners for a single instance attribute is one joined federate.
6	Federate	The ruleset for the data and its exchange within each federate is given in the SOM, which follows the restrictions of the OMT.
7	Federate	The SOM determines the principles for the updating and reflecting of instance attributes and the sending and receiving of the interactions for each federate.
8	Federate	The SOM determines the principles for the changing of the ownerships of the instance attributes.
9	Federate	The SOM determines the principles for the conditions, when the federate provides updates of its instance attributes.
10	Federate	The time management of the federates must be implemented in a way, that allows the interoperability between the federates despite of the differences in their current local time.

2.3.3 Federate interfaces

Next, we address the second standard of the three main HLA specifications: IEEE 1516.1–2010 Federate interface specification. The purpose of this standard is to define the common interfaces and services that the federates can use within the federation to succeed in the federation execution properly [12]. The API provided in this standard is implemented between the federates and the RTI, and it consists of six basic groups with an additional support service group.

The first basic group of services is the federation management, which provides the tools for the managing of the federation execution. The actions, which can be performed are the creation of the federation execution, modification and dynamic control of it including joining and resigning of the federates, and the deletion of it. Federates can only join to the federation, when the federation execution has started, and the federate is connected to the RTI. When the federate has joined to the federation execution, the RTI can support the federate by its provided services. This state is called *supporting joined federates* state, and all the other basic services, which are stated later in this chapter, are happening within this normal operational state of the federation execution. After having done its part in the federation execution, federate can leave from it via *resign federation execution* and *disconnect* services. If there are not any federates to serve for the RTI, the federation execution transfers to *no joined federates* state. In addition, if there are not any connected federates, the federation execution is destroyed eventually, and it moves to *federation execution does not exist* state. However, new federation execution can be started, if a new federate connects to the RTI, and the joining and serving process starts again, similarly as before. The life cycle of the federation execution process is presented in Figure 2.

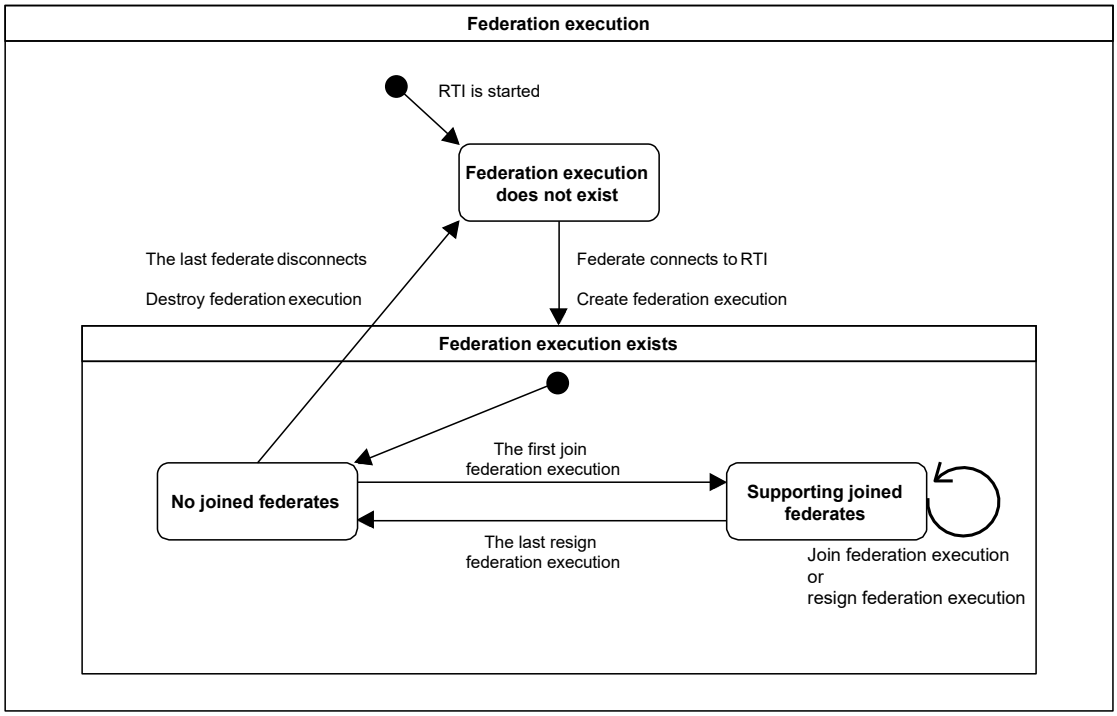

Figure 2: *Federation execution process diagram, adapted from [12] © 2010 IEEE.*

Furthermore, it is noteworthy that a single federate can be a part of more than one federation, and within a federation there can be multiple federates. Also, during the initialization part, when the federate is connecting to the RTI via a *connect* service, a callback model is specified as immediate or evoked. A callback means a service which is initiated by the RTI. For example, a *reflect attribute values* service, inside object management services, is one important callback service. Thus, if the federate has specified the callback model as immediate, all instance attribute updates that the federate has subscribed to, are reflected to this federate immediately by the RTI. If the callback model would be evoked, the attribute updates are reflected when the subscribed federate asks for new changes via *evoke callback* or *evoke multiple callbacks* services. Lastly, all the callbacks can be disabled by using a *disable callbacks* service found inside support services. Then the RTI will not start any callback service with this federate regardless of the callback model. The callbacks can be restored by calling an *enable callbacks* service. By using these support services, additional guard mechanisms and customization of the federates can be performed. [12]

The next basic group for services is the Declaration management (DM) service. Like the federation management, the DM services are closely related to the initialization phase of the federation as the joined federates declare the type of information they intend to produce and receive in the federation. These declarations happen before any instance attribute update, interaction, or object instance discovery between the federates can occur. Each federate declares only the relevant information that they are interested in or

they will produce. Thus, other federates can identify, in which federate are the usable and notable information for their usage. The information can be the whole object, or some of the instance attributes of the object. Also, the declarations must comply with the federate-specific SOM and the federation wide FOM to allow a successful federation execution. [7][12]

Within the DM, there are several services, including but not limited to a *publish/unpublish object class attributes* service, a *subscribe/unsubscribe object class attributes* service and equivalent services for the interaction classes. So, one example of a simplified declaration process when a new federate A joins a federation execution, could progress as follows: a federate A joins the federation execution, and it declares the instance attribute X that it will provide via *publish object class attribute* service. After that, a federate B which already belongs to the federation, is interested of the instance attribute X that the federate A owns. Hence, the federate B subscribes only to this instance attribute by sending a *subscribe object class attribute X* message to the RTI. Now, the RTI knows to relay any update in the instance attribute X to the federate B, or any other federate, which has subscribed to the updates of X. After some time, the federate B does not need updates from attribute X, and the federate B uses the *unsubscribe object class attribute* service. The RTI informs the federate A that no one is listening to the updates of instance attribute X anymore via *stop registration for object class* callback service. However, this message is only advisory, so the federate A can continue to send the updates to the RTI. This example declaration process is seen in the Figure 3. There are also some services in the figure that are considered in more detail in the next paragraph about object management.

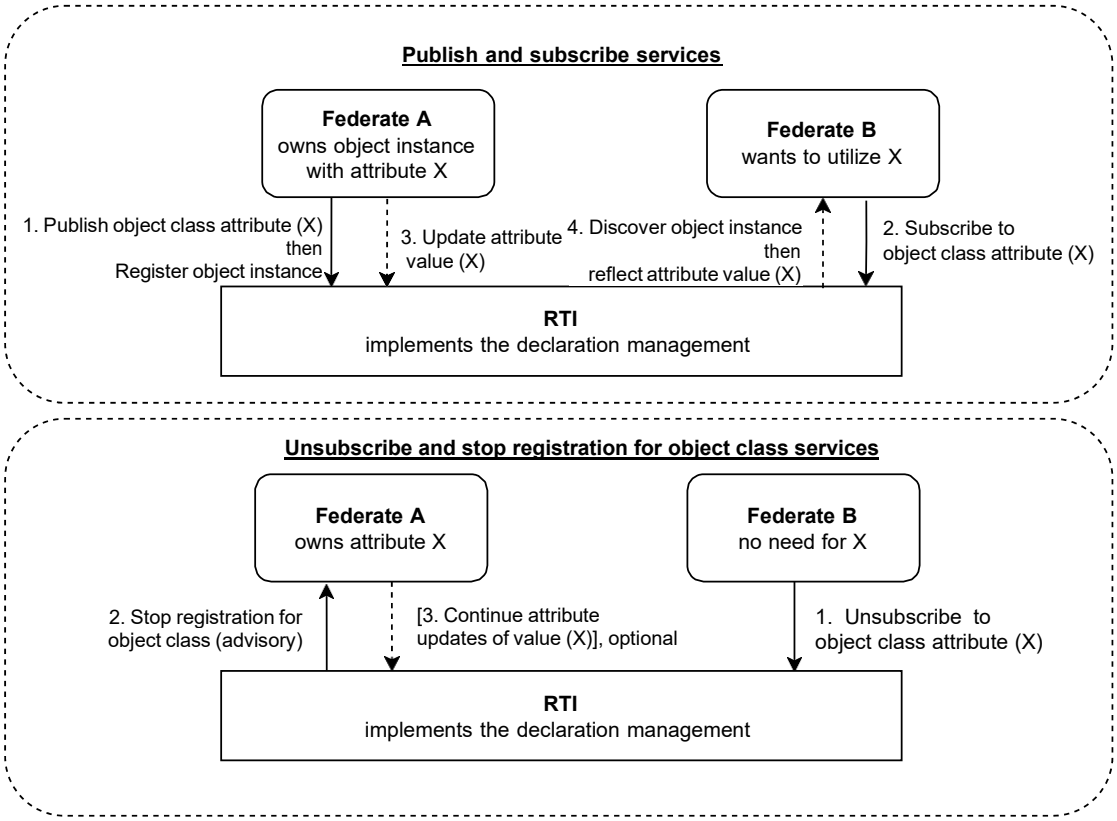

Figure 3: *Example of declaration and object class registration process [12].*

The object management complements the DM services, and it provides more details to finding, registration and deletion of object instances [12]. Also, these methods play an important part within the federation's exchange of data and information. Before any information exchange of object instances or the instance attributes can happen after the declaration, the object instance must be registered to the RTI. The registration of an object instance is generated with a *register object instance* service, which is invoked by the federate. If the object class is defined in the FOM and there are not any other exceptions, such as instance name duplicates or an unpublished object class, the object instance will be registered. When registered, the object instance can be discovered by the other federates. The discovery is initiated by the RTI, and it is provided to the federates that have informed the interest of following to this object class or its attribute via a *discover object instance callback* service. The registration and the discover process is presented in the Figure 3. When the object instance is discovered, the updates of their instance attributes are forwarded to the other federates with the help of an *update attribute value* service and a *reflect attribute values* callback service.

As the data exchange between the federates happen when a joined federate updates its instance attributes and the RTI reflects them forward, there must be a maximum of one federate responsible for each instance attribute to make the system work efficiently. The

federate that is responsible for updating the value of the instance attribute and turning the updates of the attribute on or off, if requested by the RTI, is called the owner of the attribute. To achieve a reliable and adaptable federation execution, the ownerships of the instance attributes can be dynamically changed, by following the interface declared in the ownership management services. [12] The change of the ownership can be initiated either by the owning joined federate or the non-owning joined federate that has requested the ownership. Both federates must mutually agree the ownership change and the new owner-candidate federate must have published the instance attribute in question in order to happen any change of ownership. There are different kinds of change operations, some of them are less complex than others, but the main difference between them is having a condition or not to make the change happen. With an *attribute ownership divestiture if wanted* service, the ownership of the instance attribute is changed, if some other federate is already attempting to acquire the instance attribute and thus fulfilling the condition. In contrast, if joined federate uses an *unconditional attribute ownership divestiture* service, the instance attribute becomes immediately unowned. Another way of directly releasing the ownership of some instance attribute is to stop publishing the class of the instance attribute. If the instance attribute becomes unowned, the RTI tries to find a new suitable owner from the federates of the federation. If not found, the attribute stays unowned and cannot be updated, which can lead to problems in the federation execution.

The main goal of time management services in the federations is to allow each federate to operate within their own time regardless of the current time at the other federates. With the help of time management services, the RTI can deliver and relay messages between the federates in a consistent and causal way. [12] Each message from the federates can be put to federation-wide timeline with the help of timestamps. If the federate produces timestamped messages, it is called a time-regulating federate. In addition, if the federate wants to receive and utilize timestamped messages, it must follow the time advancement procedures of the federation and is called a time-constrained federate. The default state of the federates is to not be a time-constraint or time-regulated federate. However, timestamps can still be added to the messages, but they are forwarded as they are without any rearranging or management. This order type of message is called a receive-order (RO) message. When the message, usually an instance attribute update or reflection of it, is forwarded in the order of the timestamps, the order type is called a time-stamp-order (TSO). The RTI will not forward the received timestamped messages, starting from the oldest one before it can guarantee that there are not older messages to be sent.

Each federate that joins in the federation gets a logical time from the RTI. The logical time represents the state of the federate's time in the federation-wide time-axis. The logical time can only advance forward, and the overall time management happens when the federates request time advances. The time advance request usually means that the federate has processed the previous messages and is ready to for the messages or events in its queue. When the time advance is triggered by completing an event, the federate is using a *next message request* service. After processing the event, the federate's logical time advances and it can move to perform the next event in its queue. The time advancement can also be triggered periodically within a certain time interval. Then, the federate uses a *time advance request* service to update its time. The time interval approach is more feasible for the not so event-driven and more passive federates. The third method for the federate time advance is to use a *flush queue request* service, where the federate wants to have all the messages in its queue immediately in TSO. However, now the RTI cannot guarantee if there are still older upcoming messages on their way to the RTI, which can lead to rollbacks and retractions. Thus, this service should not be used repeatedly.

The last basic service in the federate interface specification is the Data distribution management (DDM) which is used to improve the efficiency of the data exchange by reducing the unnecessary traffic between the federates [12]. In the DM, the relevance of the data was presented in an object class level. This means that the federates know, which federate updates data for the object classes it publishes. But now in the DDM, the level of abstraction is one step deeper in the class instance level. Hence, the federates can specify which object instances are relevant and irrelevant leading to more efficient communication and decreasing the load of the federation.

The relevance of the instance attributes can be defined within specific bounds of region space by the federates. The producers, or the federates sending information, have declared their own upper and lower bounds of the space, the update region. Whereas the consumers or the receiving federates have specified their own subscription region. If these regions overlap, there are instance attributes that are relevant for both producer and consumer, and they can only focus on receiving or transmitting these instance attributes while the RTI filters the irrelevant ones. [12]

Finally, in addition to the basic services, support services provide miscellaneous supplementary services that the federates can also utilize. These services include name-to-handle and handle-to-name transformations where objects classes can be called by their unique names, or the name of the object class can be retrieved if given its unique handle. Also, advisory switches can be created which are triggers that send notification

to federates in question when a specified condition happens. In addition, some modifications to the regions of the DDM can be made from the support services. Lastly, the mode of the callback service, evoked or immediate, or the way of receiving the instance attribute updates can be set via a *specified callback* service within the support services. [12]

2.3.4 Object model template (OMT)

The final specification of the three interrelated HLA documents is the IEEE 1516.2–2010 object model template. This specification defines the format and the syntax for the presentation of the HLA objects. [9] However, the content of the objects is not considered to preserve the nature of the standard as open and general to all simulation environments. Thus, the contents of the objects can vary regarding of the purpose of the federation and they are specified in each federations' FOMs and SOMs. Nevertheless, the OMT enables a common structure of object models, which helps the federates and the RTI to coordinate together by being able to "speak the same language". Also, the HLA objects specified in this document are the necessary tools that describe a single federate or federation's capabilities in a standardized format through their FOMs and SOMs. With this information, other federations and federates know exactly what type of information or benefits these potentially joining federates or other federations could bring into their federation.

The object model used in HLA is similar to object-oriented programming (OOP), but there a few key differences: in HLA the objects are defined only by the object's instance attributes and their values, for example a vehicle object having an instance attribute fuel with value of 100. Whereas in OOP, the objects can have data members and methods which can be called directly by the other objects. Contrary to OOP, HLA objects do not communicate directly with each other; it is the federates that communicate via the HLA services, for instance updating or reflecting the instance attribute values or sending interactions as stated earlier. Also, the owning or the update responsibility of the instance attributes can be divided more freely in HLA to different federates while in OOP the updates happen more locally and be stricter described with private, protected and public types of values. [9] In addition to object classes, HLA has also an alternative class type, an interaction class. This class describes the interactions, or the relationships, between the object classes with separate parameters without the need of creating an object instance. [7]

The HLA object models are formed from several interrelated components that specify the representation form of the information it provides. For example, the first component

object model identification table consists of two columns: category and description. The category being the name of a parameter and the description a text which briefly describes the parameter. The category could be purpose, type or name of the object model. The corresponding descriptions would be then: to model a vehicle federate, SOM and vehicle, respectively. The main purpose of this component is to document the object models for possible future development purposes or troubleshoots. The object model identification table and other components that must be addressed are presented in the Table 2.

Table 2: *OMT components and their descriptions [9].*

Component	Brief description of component.
Object model identification table	Documents the purpose of the object model and other relevant parameters that identify it from the other objects.
Object class structure table	Describes the relations of the classes and subclasses or the inheritance architectures.
Attribute table	Specifies how the attributes of the objects are presented.
Parameter table	Specifies how the parameters of the interaction classes are presented.
Dimension table	Specifies the dimensions that are utilized in the DDM process of filtering irrelevant instance attributes.
Time representation table	Specifies how the time values are represented.
User-supplied tag table	Specifies how the additional tags are represented with certain HLA services.
Synchronization table	Specifies how the datatypes of the HLA synchronization service are represented.
Transportation type table	Specifies the transportation mechanisms. Can be reliable or best effort.
Update rate table	Specifies the update rate for the information.
Switches table	Specifies the initial parameter settings for the use of the RTI.
Datatype table	Specifies details of data representation.
Notes table	Specifies some additional explanation for OMT table item.

The OMT components as such are not sufficient for providing a fluent simulation interoperation. Thus, the OMT also defines a collected set of tables which accurately describe the semantics of the classes, attributes and parameters. In other words, explanation for each class, attribute, or parameter in FOM and SOM is given in order to document the federations or federates' capabilities. This section of the OMT is called the FOM/SOM lexicon. As mentioned earlier, there are several simple tables that form the FOM/SOM lexicon: the object class definition table and attribute definition table are closely related to each other. In the object class definition table, there are two columns: one describes the name of the object class, and the other column describes the object class briefly. The attribute definition class has three columns: the name of the attribute, the object class it belongs to, and brief description of the attribute. Similarly, also the interaction classes and interaction class parameters are explained in the interaction class definition table and in the parameter definition table. With the help of these semantics, OMT components and some conformance rules presented in the OMT specification, a more effortless simulation interoperability can be achieved, which leads to overall better results in the federation executions. [9] Now, the three main specifications of the HLA standard have been presented, and in the next chapter we consider data transfer and CDS and how the HLA standard is utilized with them.

3. TRANSFER OF DATA AND GATEWAYS

The data transfer from one place to another consists coarsely of three parts: the transmitter, transfer media and the receiver. The transmitter produces the data to be transmitted and the receiver is the destination of the data. In between is the transfer media or the channel which can be wireless or wired. In modern digital communication systems, the data is mostly digital consisting of zeros and ones. When the zeros and ones, or the bits, are ordered in a certain known way, information can be interpreted and transmitted digitally to others. Also, conversion from digital to analog and vice versa enables the transmission and the receiving of analog waveforms, also known as wireless communication. Next, we are focusing more on the wired transmission where the data moves usually in certain repeating structures called the data packets.

3.1 Data packet structure

Data packets are typically formed from two parts: the header part and the payload part. The payload part contains the information that is the actual information to be sent, such as a message or the updated values of some parameters. In contrast, the header part contains redundant data for the information, but necessary for the success of the transmission. An example header could contain some preamble sequence, the destination of the receiver, the source of the sender and the size of the payload. The size of the whole data packet is determined mostly by the size of the payload, while the header has only a small part of it. This ratio of a header and a payload should weight the payload part to be able to maximize the throughput of the transfer system.

Another way of improving the throughput of the data packets is to compress parts of it, which reduces the total number of transmitted bits in a data packet for the same information. If the compression can be reversed and no information is lost the compression is lossless. However, if some parts of the data are not recoverable after the compression, the compression method is lossy. [13] The lossless method should be used when loss of information is not tolerated in any form. This is evident in the case of transferring and compressing written documents, where the integrity of the information should remain unchanged. One example of a lossless compression is using the standardized compression method ZIP [14]. The lossy methods can be used when the loss of information is acceptable, for example converting images to smaller formats to save memory, such as using the original JPEG format [15].

3.2 Secure gateways

Gateway can be defined as a device which connects several different purpose networks together [16]. The separation of two different networks is an important concept when considering the information security point of view. Usually, if two computers are in the same network, they can find and then communicate with each other, which is a preferable thing if both of the parties are co-operating and the common goal is the same. However, if the other party is not aware of the other, the communication between the computers in the same network can be used for harmful purposes, such as eavesdropping, capturing and modifying messages and accessing sensitive data. To avoid the unwanted traffic and operators, gateways are used. The gateways can be found from the homes of the people or in the corporates' offices where several networks with different purposes are connected to form private networks. Thus, the people from the outside cannot access these networks without permission or authorization.

Another operating principle for the gateways is to act as a data filter that allows, blocks, or modifies certain data that passes through it. With this type of secure gateway solution, different networks can be connected and communicate together even though the other one can contain more sensitive information than the other. The sensitive part of the data is stripped or changed within the gateway before it is relayed forward to the other networks. At the same time, data can pass unfiltered or filtered if needed to the stricter and more secure network. One example of these secure gateways is the Insta DefSec's product Cross Domain Solution (CDS) which is covered in more detail in the following chapter. Also, as a note regarding the remainder part of this thesis, the abbreviation CDS refers to this product as it can also be a synonym for a general secure gateway.

3.3 Cross Domain Solution (CDS)

CDS is a secure gateway solution developed by Insta DefSec that allows message filtering and modification between two differing security level networks. The solution consists mainly of three parts: The user interface tool, the rule engine and the logger. The overall presentation of the system diagram is shown in Figure 4.

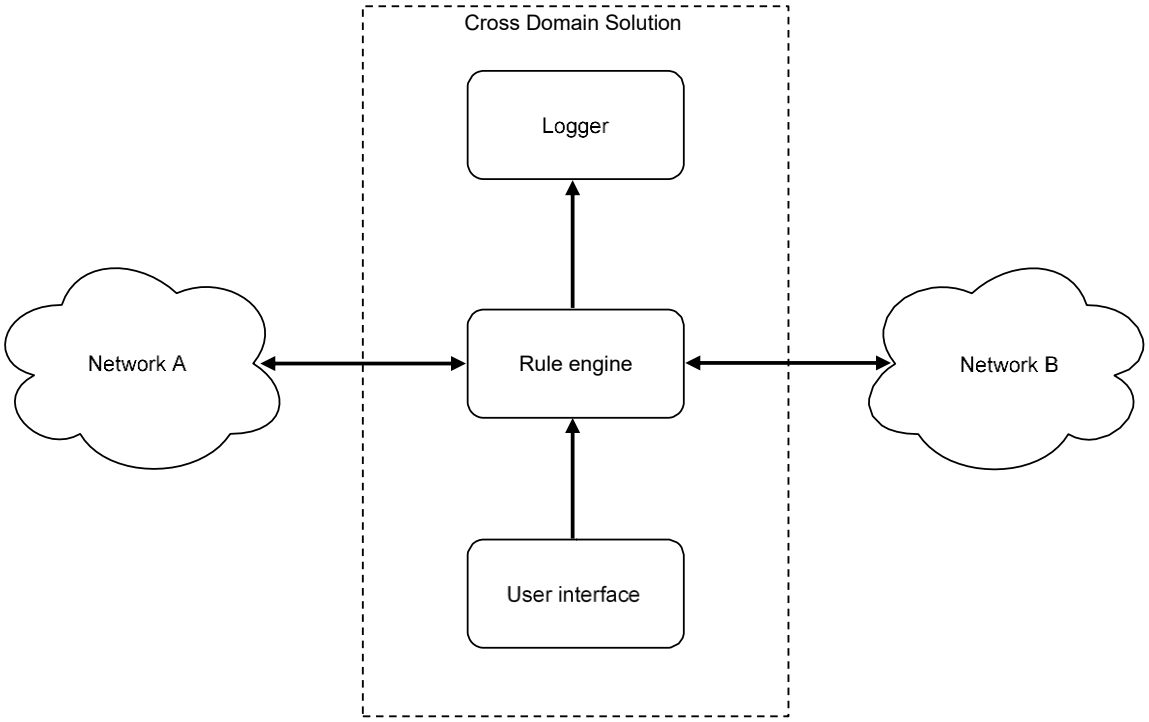

Figure 4: *The system diagram of CDS [17].*

The CDS user interface is an application where users can define the rulesets of the rule engine. With clear and coherent symbols, buttons and input fields, the user interface enables the usage and modification of CDS rulesets for people with less technical expertise. This reduces the need for help from the developers or from the system administrators when reconfiguring the rulesets for different purposes. The ruleset is then transmitted to the rule engine which implements the ruleset in practice. In addition to outputting the data in the desired format of the ruleset, the rule engine has a logger output. If the data coming to the rule engine is modified or filtered, the corresponding actions are logged. Thus, the main purpose of logging is to keep track of the history of the modified data, but it can also be used for debugging the system if it is not operating as expected. [17]

3.3.1 CDS rulesets, rules and conditions

The CDS ruleset forms three layers of hierarchical levels: the top level are the different rulesets which are given by the specific rules. To get the individual rules that define the ruleset, conditions and corresponding operation are given for each rule. The condition is the triggering element that activates the rule. The triggering can happen when some specific value of the received data is equal to, not equal to, greater or less than, greater or less equal than the threshold value for the condition. After the triggering, the corresponding operation is executed for the data packet in processing. The available

operations are blocking, passing, logging and transformation. With the blocking and passing operation, the received data packet can be filtered completely, or passed without any altering. Also, the logging operation lets the packet through, but with a customized logging message. The transformation operation needs an additional parameter for the replacement of the specific part of the packet.

As each rule can consist of many conditions, it requires all of them to fulfill the rule and make the operation happen. In other words, the conditions of the rules have AND logical operator relationship. However, the individual rules which are the combinations of conditions and operations have mutual relationship of logical operator OR. This means that, the triggering of a single rule does not require the fulfillment of the other ones within the ruleset. Thus, the overall rulesets can be large and customized for various purposes.

3.3.2 The functioning of the rule engine and HLA integration

After the ruleset is defined, it can be imported to the rule engine which then implements the rules of the ruleset in practice. The rule engine uses the imported ruleset directly as a generic form for all the protocols it must handle. Thus, some specific protocol-to-general form adapters are needed before applying the rules, to handle data from the different protocols. Currently CDS can support data from protocols that have certain structure in their messages. The two these kinds of protocols that CDS supports, are the HLA and the All Purpose Structured Eurocontrol Surveillance Information Exchange (ASTERIX) protocol. Both protocols have a simple hierarchical structure that can be mapped similarly. [13] For example, the hierarchical dependencies of the object and the interaction classes of the HLA are simple to present as a general tree structure with root and nodes. The object class hierarchy of the RPR FOM 2.0, considering the root node HLAobjectRoot and only its BaseEntity branch with its subclasses, are presented in the Figure 5. The RPR FOM 2.0 also contains several other object class branches, such as EmbeddedSystem and EnvironmentObject that are formed from the HLAobjectRoot class. Due to vast number of these other classes, they are not visible in the Figure 5.

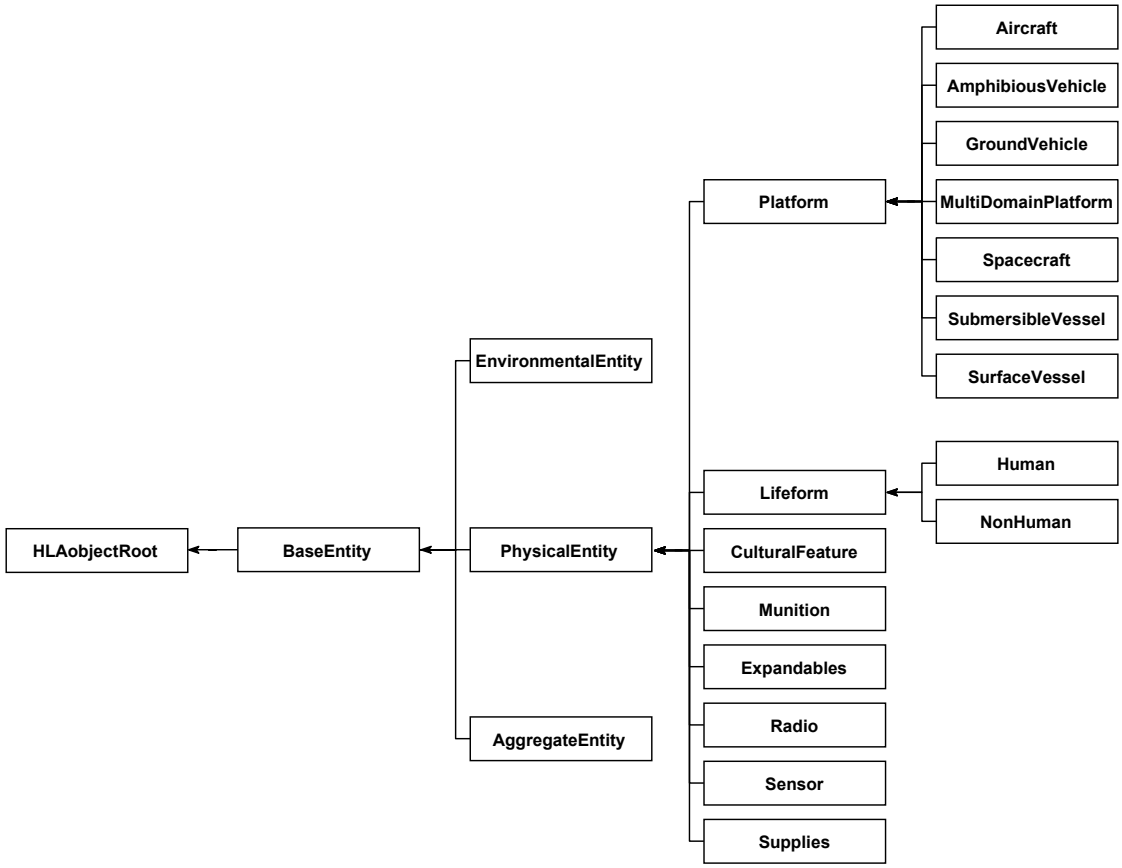

Figure 5: *RPR FOM 2.0 BaseEntity branch object class and its subclasses [11].*

The RPR FOM 2.0 also contains the definitions for the interaction classes that are available for the federations which utilize this FOM. Some of the interaction classes that are based on the HLAinteractionRoot are Collision, WeaponFire and EnvironmentObjectTransaction. All the other interaction classes and their subclasses, and the complete hierarchy of the object classes under the HLAobjectRoot can be found from the paper defining the RPR FOM 2.0. [11]

As the object and the interaction classes relationships are straightforward to follow, also the values of the instance attributes of the object classes and the parameters of the interaction classes can be mapped effectively. When the owner of the attribute or the parameter is known, the corresponding key-value pair is easy to retrieve from similar mapping of a tree structure.

After the conversion to a general form in the adapter, the rule engine can parse the received message for the comparison between the rules. If the rule engine matches a rule to the message, the corresponding operation is executed to the message. An example ruleset that filters the HLA messages belonging to a class Platform is presented in the Figure 6. The key specifies the possible object classes of the message, value is

the parameter that the key is compared with the operator, and if the condition is valid, then the operation is executed.

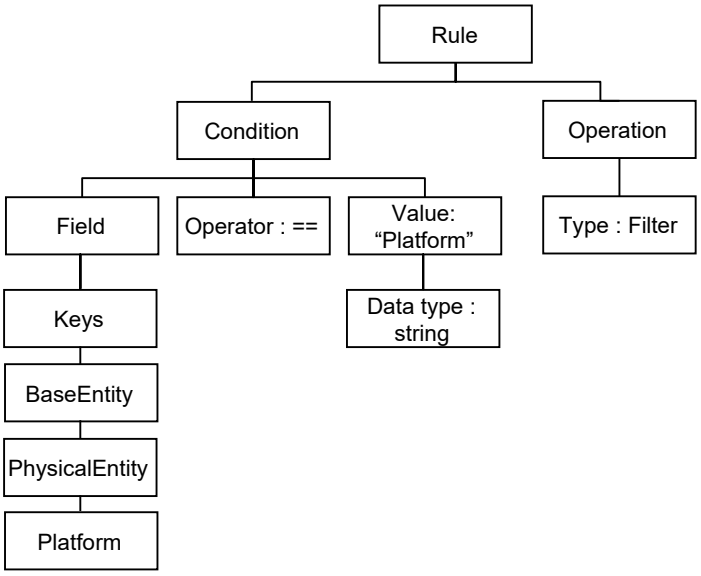

Figure 6: *Example ruleset model for HLA. The example filters all HLA messages from the object class Platform and its subclasses. Note, all possible object classes are not shown for presentational purposes [17].*

If the operation to the message is executed, the operation is saved by the logger and the operated message in question is transmitted onward to the following network or filtered out depending on the type of the operation. Then, the next message is compared to the same rules of the ruleset, and the process continues as long as there is data to be forwarded between the networks.

4. PERFORMANCE OF SIMULATIONS

The performance requirements for the well-functioning real-time simulation systems should be known in order to find out if the CDS is an applicable solution as a secure gateway between multiple real-time simulation networks. The following findings from the literature, the experts' opinions and the reference measurement should act as benchmarks which give the approximate region where the end results of the measurements should be located at.

4.1 Performance requirements for real-time simulators

The existing performance requirements of real-time simulator systems were studied with the help of different sources, including technical reports and specifications from RTI manufacturers, other scientific publications and theses, inquiries from the experts of the industry and from the reference measurements of commercial flight simulator software Prepar3D.

The performance metrics that were selected for the evaluation of the performance of the simulator system were the latency and the throughput. These performance metrics form the requirements that the correctly functioning real-time simulation networks should follow in all cases, including a configuration with a CDS solution involved. Thus, the time taken for each update and the amount of data that can be transferred, should follow the limits presented in the upcoming chapters regarding the studies and research of latency and throughput, and the views of the experts of the industry of real-time simulator systems.

4.1.1 Latency

Latency is a metric of time which presents how much time it takes for a single data packet to get from the transmitter to the receiver. There is always some amount of latency in data transfer due to the constraint of the speed of the light, but on top of that, the latency increases when the data packets are processed in various nodes and processors. Thus, the latency is usually an unwanted phenomenon, but it can be tolerated to a certain limit, which depends on the use case of the data transfer system. With real-time systems, two general categories can be found: the systems having a hard or soft deadline [18]. In the other end are the soft deadline systems where missing the deadline for receiving the information on time is not desirable but it can happen. For example, some non-critical

measurement system measuring the highest temperature of the day, but it had some major delays in the measurements because of the malfunction and restarting of the systems. The lack of receiving the data is not preferable, but it will not form any potential risks to others. In contrast to soft deadline systems, there are the hard deadline systems that do not tolerate any delays for the deadline and if the latency exceeds certain limit, it can have critical consequences for the system or its users. Hard deadline systems can be monitoring systems of critical infrastructure or real-time LVC systems. As an example, a system monitoring the state of a nuclear power plant cannot tolerate any major delays in its measurements if something seems to be not working properly. Next, we are focusing on studies of the latency requirements on different real-time simulation systems, focusing mainly on those that consider the standard of HLA.

Firstly, there are not any clear indications of the latency requirements of the HLA in its three main specifications, the framework and the rules, interface specification or in the OMT. However, there is a small note or an example in the OMT that sets a coarse scale for what the requirements could be: the example of an update rate table that defines the optional maximum update rate within federation for the owners of the attributes. The table defines that a high update rate is 30 Hz, medium is 5.0 Hz and low is 0.2 Hz [9]. Therefore, if the update rate is 5.0 Hz, it means that an update is sent 5 times in 1 second, or every 0.2 seconds. The corresponding latency limits for the high, medium and low update rates are then 33.3 ms, 200 ms and 5000 ms, respectively. Hence, it is feasible to assume that in the case of a real-time simulation system, the latency of the update should be lower than the update rate, to enable the receiving of the data packet before its next update. If the latency is higher than the rate of the updates, the most recent data is not utilized in the latest update, and the simulation presents outdated information. This can produce problems, especially if the delayed information concerns the parameters of an important entity. With misleading parameters, the right decisions will take up time or can lead to even wrong conclusions with serious consequences. Thus, a real-time LVC system can be considered more as a hard deadline system than a soft deadline system.

Similar range of latency results, in the order of 30–100 ms, were requested by a paper from the RTI manufacturer MAK Technologies for the limits of real-time simulation before the feeling of the real-time interactivity becomes intolerable for the operator [19]. Also, it was mentioned in a master’s thesis, that a rate of 10 frames per second is a minimum acceptable rate for a well working real-time simulation system [20]. As the latency increases, the updates will be received later. If the simulation updates its view only when there are new updates, the immersion of real-time simulation can disappear if the updates of the simulation are delayed by the increasing latencies. However, the tolerable

latencies depend on the purpose of use of the simulation. As said earlier, real-time and immersive simulations such as flight simulators need low latencies, but some other simulations could manage with higher latencies.

Overall, the key point from all these results would be that the upper limit for the latency should be approximately 100 ms although lower latencies are expected from the measurements considering real-time simulation network with and without the CDS attached utilizing the HLA. This requirement is one of the performance metrics that are evaluated in the final measurements where the effect of the CDS is examined.

4.1.2 Reference measurement of a commercial simulator system

As a part of defining the estimate of the throughput benchmark for the real-time simulator systems and to familiarize with a real-world simulation implementation, a reference measurement for a commercial simulator system was made. The simulator of interest was the flight simulator Prepar3D made by the Lockheed Martin [21]. The simulator in question was chosen due to its presumable high-performance requirements and the capability of measuring them with ease via utilizing built-in API.

The reference measurements were two parted due to the different main protocol of the Prepar3D which was not HLA. The main protocol used in the Prepar3D was DIS, the predecessor of the HLA. Thus, the first measurements were done by capturing the DIS data traffic directly from the API with the help of Wireshark [22]. The other measurement required a DIS protocol conversion to HLA by using a third-party software VR-Exchange to have an equivalent measurement with HLA [23]. However, the Wireshark does not support HLA messages, so the converted HLA traffic is captured by the test bench software that is used also in the final measurements of this thesis. The test bench logs each received HLA message with timestamps for further analysis.

The set-up for both these measurements were similar: one aircraft entity in the flight simulator flying straight and level, which updates its location and other parameters with approximately constant rate to the API. The update rate is determined by taking the difference between two consecutive timestamps for all the flying entity-updates and then averaging them. The only major difference between these two measurements is the conversion to the HLA and the data capturing software. Thus, the conversion block might add some additional delay to the system. The results of the reference measurements are presented in the Table 3.

Table 3: *The results of the reference measurements of a commercial flight simulator.*

Reference	Protocol	Update interval (ms)		Samples
Type	-	Average	Std	-
Measurement 1	DIS	94.4	83.7	2130
Measurement 2	HLA	111	197	4505

The measured update interval was around 100 ms for both DIS and HLA messages. The minor difference between these two could result from the different message size and structure of the HLA and DIS protocols. Also, as mentioned earlier, the conversion from DIS to HLA can add some delay, which could explain the slightly bigger update interval of the HLA. In addition, the standard deviations of the update interval measurements were in the same magnitude as the averages, which means that the update intervals varied at different instances of time remarkably even though the aircraft flew at a constant speed and altitude and did not do any additional maneuvers.

4.1.3 Throughput

The second performance metric that helps to evaluate the suitability of the CDS to the real-time simulation networks is the throughput. Generally, it determines the amount of data bytes that can be transmitted from the transmitter to the receiver in certain time interval, usually in one second. However, the throughput can also be presented as updates/s or entities/s, which is a more user-friendly approach to present the throughput information. To maximize the potential capacity of the system, the supported throughput should be high. This reduces the queueing of the data if the amount of the data to transfer increases suddenly. The reduction of the queues also decreases the latency, which is also beneficial for the simulation systems in general. As the simulation systems tend to produce a lot of data, high throughput requirements are essential for these systems. Especially with the real-time simulation systems, the throughput must be high enough to manage the load all the time even if there are sudden changes in the number of entities or other types of events that increase the traffic significantly.

Similarly, as in the case of latency, the HLA specifications do not provide any defined number for the throughput requirement of the HLA systems. However, the lack of this definition follows the idea of the HLA being as general and versatile as possible. Same throughput rates cannot be promised to be achieved with highly complex multiple

federation connecting simulation systems as with simple single federation and federate system. Furthermore, suitable studies and papers were not found from the literature addressing the throughput requirements of the real-time simulator systems, which lead to acquiring the appropriate benchmarks via interviewing the experts of the industry for their opinion. With the help of their insight and from the data gathered from the commercial real-time flight simulator software, also the throughput requirement could be established.

A brief interview in the form of a survey, was sent to four different experts specialized in this field [25]. The purpose of the survey was to find out what would be the maximum number for simultaneous entities that would still be realistically presented and updated in the same simulation or federation. The expert A mentioned that the range of 200–250 would be a typical upper range, but even higher amounts could be temporarily supported. However, a constant exceeding would degrade the overall performance and should be avoided with the help of possible countermeasures. Similarly, the expert B concluded that the upper range would be around 150 flying entities with additional 30 ground-based entities. However, this could be achieved with a significantly smaller update rate where all the entities would be updated after every 6 seconds. The view of the expert C also contained a separate estimate between simultaneous flying and ground-based entities. The theoretical upper limit for the flying entities was 210 while additionally 100 ground-based entities could be simultaneously supported. Furthermore, the same expert gave a more realistic view where the number of flying entities would be a half from the theoretical, but the ground-based entities would remain the same leading to a more practical estimate totaling of 205 simultaneous entities. The last response was from the expert D, who gave the widest range of 20–200, which was based on different simulation scenarios where the expert had been involved in.

When considering all the answers from the interviews, the ranges varied above and below 200 entities. According to experts A and C, the simultaneous entities above 200 could be interpreted as more theoretical or momentarily states that could possibly be achieved only temporarily. Also, the experts B and C highlighted the separate numbers for the flying and ground based entities. Therefore, it could be presumed that the different types of entities can stress the simulation system differently. Mainly the ground-based systems can remain stationary, which does not necessary trigger entity updates when nothing is changed in the parameters of the stationary ground entity. In contrast, there are the flying entities that produce continuous updates which cannot be stationary as their ground-based counterparts. Thus, the number of maximum simultaneous entities would be lower if all the entities would be flying compared to flying and ground-based

entities combined. However, the RPR FOM V2 defines aircraft and ground vehicle entities practically with the same attributes, so the division between these different entities is not remarkable, and they are considered as the same in the final measurements and in the throughput requirements.

Furthermore, the more moderate estimates proposed by the experts D and B were in the range of 20–200 and 180. So, the 200 simultaneous entities were now in the upper range of the estimate. However, the view of expert D was based on in the involvement in various scenarios, which could have a very different context in them. For example, some of the simulations could test only the performance capabilities whereas others could focus on other scenarios where the number of 20 simultaneous entities would be enough. Thus, the upper range of this estimate would fit better with the views and outlines of the other experts.

Lastly, the view of 180 simultaneous entities from the expert B is similar to other estimates, but it was mentioned that it could be achieved with a lower update rate of after every sixth second. If the update rate is increased to 10 times in a second, which was measured with the commercial flight simulator, there could be major differences in the capability to support the increasing number of the required updates. With 180 entities that update after every 6 seconds, the required throughput is 180 entities divided by the 6 seconds leading to throughput of 30 entities/s. This is much less than the corresponding throughput which is calculated with the update rate of after every tenth of a second. With the update after every tenth of a second and with 180 simultaneous entities, the throughput would be 1800 entities/s, which is 60 times bigger estimate than the previous one. These results would implicate that the estimate of the expert B would be lower if the required update rate would be higher, as it will be presumed in the final measurements of this thesis.

As a conclusion, the approximate maximum number of simultaneous entities could be estimated by calculating the weighted average from the views of the experts. As discussed in the previous paragraphs, the experts had some variations in their views, and the preconditions were not the same. Thus, the estimates of the experts have different weights for the average regarding their similarities to the test set-up of this thesis. The expert A and C both have a weight of 30 %, whereas the expert B and D both have a weight of 20 %, meaning that the views of experts A and C have more influence on the throughput requirement. These weights and the estimates result in a weighted average of 187 simultaneous entities when considering the different views of the interviewed experts. However, there can be variation in both directions, and this is only

an approximate result combined on the grounds of four different opinions with weights based solely on the interviews.

By knowing the update rate and the typical number of simultaneous entities, the final throughput benchmark could be calculated with a simple multiplication calculation. The reference measurement described the results for the update rate that was around 10 Hz in the two measurement sets, which means that every entity updates 10 times in a one second. In addition, the view from the experts of the field was that the maximum number of simultaneous entities in a realistic scenario would be approximately 187 entities. To conclude the chapter of the throughput, the benchmark for the throughput should be around 1870 entities/s when considering the update interval of the entities and the maximum number of them being simulated at the same time within the simulation.

5. IMPLEMENTATION OF PERFORMANCE MEASUREMENTS

The evaluation of the effect of the CDS to the real-time simulation network was measured with the help of a test bench software, message forwarding RTIs, CDS and VR-Exchange software. The role of the test bench software was to create the federations and federates that communicate with each other by updating the object attributes and interaction parameters. The RTIs would transfer the messages which go through the CDS or the VR-Exchange if the CDS is not attached. After the gateway and RTIs, the HLA messages are received in the other test bench which act as a receiving federate. Next, a more detailed look of the measurement system is presented to find out the basic functioning mechanisms. Also, the final measurement process is described in detail to enable the repeatability of the process.

5.1 Set-up of the system

The physical measurement set-up is formed by connecting three different laptops together with Ethernet cables. The exact configuration and the connections are shown in the Figure 7.

Figure 7: *The physical configuration of the measurement set-up.*

The two most outermost laptops in the configuration run in the Windows environment whereas the middle one operates in the Linux/Unix platform. The middle laptop has two network interfaces, and it works as a connecting element between the two separate networks it is connected to. The Ethernet connector cables are crossover cables, so any external switches are not needed within the configuration. There are four different software that are utilized in this configuration: the test bench, MAK RTI 4.2 made by the MAK Technologies, CDS and VR-Exchange. The laptops with the Windows operating system contain the test bench and RTI software while the CDS and VR-Exchange are

executed in the connecting laptop in Linux/Unix platform. The layout of these programs in the configuration are presented in the diagram in Figure 8.

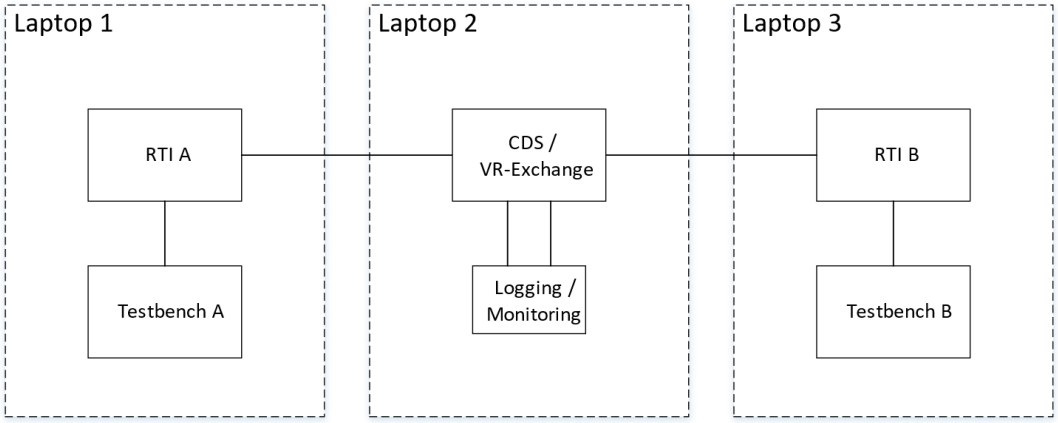

Figure 8: *The layout of the software within the configuration.*

Next, the purpose of each software in the measurements are discussed in more detail to understand their role and location in the configuration better.

The test bench software is a simple program created with JAVA, which is programmed to test the performance requirements and the capabilities of the CDS for the needs of this thesis. After the start up, it connects to the RTI, creates the federation execution, and joins to the federation as a one federate, similarly as the creation of the federation execution is presented in the Figure 2. After the joining, the federate registers the object instances and publishes the object class attributes that it will update during the measurements. Also, the necessary subscriptions are performed to the interaction classes, which act as responses for receiving the updates from the receiver end. These procedures follow the Figure 3 which presents the general steps for enabling the communication between the connected federates.

The same test bench program can be configured as a transmitter which updates the object classes, but also as a receiver which receives the transmitted updates and sends the acknowledgements of the messages with the help of the interaction classes. The operating mode is selected after the program is started to allow the usage of this program within both ends of the measurement system without any major reconfigurations. In addition, the mode for the measurement set-up is chosen due to the different nature of the two set-ups: the first set-up will be measured with a single entity that updates its attributes whereas the second one has multiple simultaneous entities that are updated. This difference between the measurement set-ups requires different configuration for the test bench in the transmitter side which can be changed from the user interface of the

test bench. The user interface also has a selection for connecting the test bench to the RTI, which makes the required procedures for the connection and the communication as described earlier, and after their successful initialization, the logic of the chosen measurement set-up starts automatically. The graphical user interface is presented in the Figure 9.

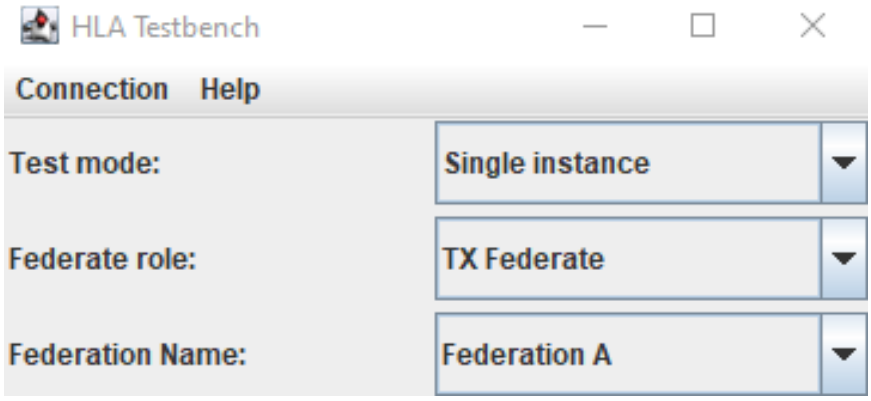

Figure 9: *The graphical user interface of the test bench software.*

The user interface of the test bench program and its components were constructed with the Swing library which is one of the standard libraries for JAVA. It enables the creation of the graphical user interface that can contain several basic components such as buttons, combo boxes, lists, labels, tables, and menu items. The main benefit of the components of the Swing is that they are all coded with JAVA, which guarantees that the graphical user interface works similarly within all the platforms it is used. [24] Another main library that was utilized in the test bench software is the hla.1516 library which implements the services that the RTI provides, or in other words, defines the methods on how to connect and communicate with the RTI and other federates. The hla.1516 library is RTI-specific because it only works with the RTI solution made by the MAK Technologies. Therefore, to be able to connect with RTIs from the other manufacturers, additional libraries should be added to the implementation. Furthermore, the newest version of the library hla.1516e was not chosen due to compatibility errors with the latest JDK11 which was used in the development of the test bench software. Also, the main functioning of the hla.1516 and hla.1516e are similar and there are not major differences that would affect the results significantly.

As said earlier, the two test bench software need a way to communicate with each other. Both test benches create messages, but the messages can only find their destination via the help of the RTI. The RTI keeps track of the needs of the federates that are joined to it and delivers the messages between them. For example, the transmitting federate wants to receive every Collision interaction which are responses from receiving the

transmitted message in the receiving federate. Thus, the transmitting federate creates a need for receiving the Collision interactions by sending a request via the Subscribe Interaction Class service. Now, the RTI knows to relay every interaction of type Collision to the transmitting federate. The transmitting federate can inquire for the possible new updates from the RTI by using the *evoke callback* or the *evoke multiple callbacks* services. If there are not any new updates that the inquiring federate is subscribed to, the RTI will inform the federate that there are not any new updates available. The same principle works also for the object instance attribute updates which are needed in the receiving federate and sent by the transmitting federate. The RTI of the receiving federate knows to relay the correct attribute updates when the receiving federate has subscribed to the right attribute updates via the *subscribe object class attribute* service.

This final measurement set-ups consists of two federations both of which are connected to their own RTIs, the RTI A and the RTI B. The configuration with two RTIs is needed to simulate the possible real-world configuration where two different networks have their own RTIs and federates. These federates and RTIs are then combined with the CDS which can filter or modify the specific messages that are specified in its ruleset. Therefore, the sensitive messages and information are not distributed outside the secure network whereas still allowing the sharing of the non-sensitive messages and information. To allow the realistic separation of the networks, the CDS is configured as the connecting block between the two networks. Hence, all the traffic will go through the CDS if data is needed to be shared between the networks. The CDS ruleset for both the measurement set-ups is the same: pass all the messages that comes to the CDS from both directions.

The computer with the CDS has also the VR-Exchange software which is used instead in the reference measurements without the CDS functionality. The VR-Exchange acts as a gateway, but it will pass all the data through it without any altering. Thus, the reference measurement without the CDS is as similar as the set-up which includes the CDS and its functioning ruleset. Thus, the effect of the CDS in the measurement set-ups can be evaluated by measuring the differences in the measurements set-ups considering both the effect on latency and the throughput.

5.2 The measuring process

The final measurement process contains two different measurement set-ups: the latency and the throughput set-ups. The latency measurement has a single object instance which updates its instance attributes in a constant frequency. The latencies of these updates are then measured with the CDS between the transmitter network and the receiver

network. The first measurement is then repeated, but the CDS is replaced with the VR-Exchange. With the help of this reference measurement, the effect of the CDS to the latency can be evaluated.

After the single object instance latency measurements, the throughput measurement with multiple object instances is executed. The throughput measurement is repeated with different amounts of updating entities, to find out the optimal conditions for the maximum throughput. The maximum throughput can be calculated when the number of received updates and the total time taken in the measurement is known. Also, the throughput measurements are repeated by replacing the CDS with the VR-Exchange. Next, the measurement procedures are presented in more detail, by introducing the initialization process and the different steps of the latency and the throughput measurements.

The latency measurements start with the initialization process of the federates which is illustrated in the Figure 10. These steps must be executed before any communication between the federates can happen.

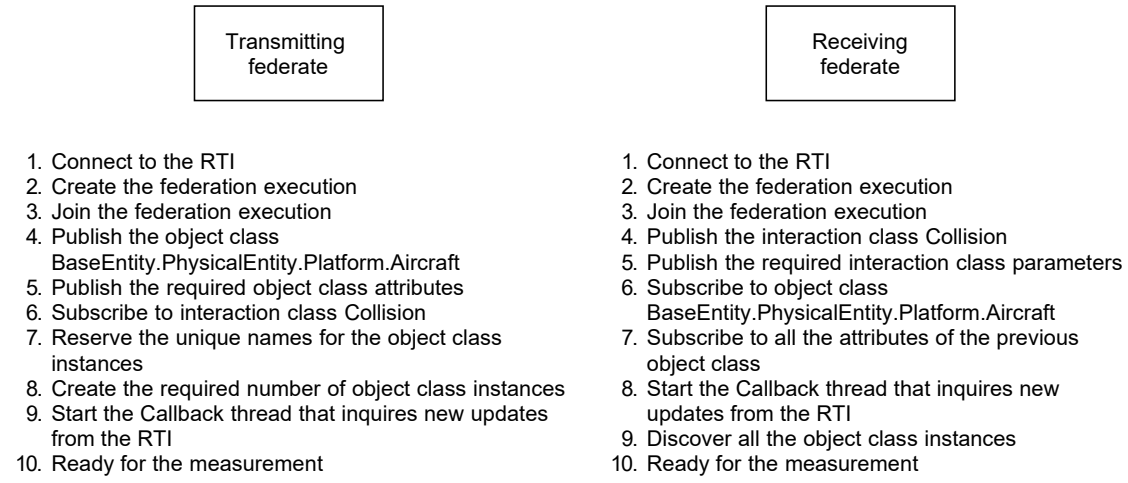

Figure 10: *The initialization process of the latency measurement.*

The first three steps in the initialization process are the same for both the transmitting and the receiving federates. First, the connection to the RTI is established by giving the ip-addresses and the port numbers of the RTI to the joining federate. After the connection is established, the federation execution is created where the connected federates can now join. The measurement system consists of two RTIs, so there are two different federations. Hence, the names of the federations must be unique and can be selected from the UI of the test bench software before the initialization process. Also, the CDS with a configured ruleset, or alternatively the VR-Exchange, must be started and connected to the RTIs to get the gateway between the RTIs work properly. The other

port of the gateway is connected to the same federation as the transmitting federate while the other end is connected to the same federation as the receiving federate.

Next, the federates publish either their object or interaction classes and their attributes and parameters which will be updated in the measurement set-up. In addition, the federates must subscribe to the interaction classes and object attributes that they wish to receive. Within these measurements, the receiving federate is only interested in the attribute updates of the object class of BaseEntity.PhysicalEntity.Platform.Aircraft, while the transmitting federate is waiting for the interactions of the receiving federate. The transmitting federate also creates the required amount of object instances, or also referred as entities, that it is going to need in the measurement set-up, which each have a unique name that can be reserved manually or generated automatically by the RTI.

To find out the existence of these unique object instances and to be able to get the newest attribute updates or interactions from the other federate, a callback to the RTI must be made. If the federate sends a callback request to the RTI, the RTI will inform the federate of the updates that have been happened to the object attributes and interactions it has subscribed to. A separate thread for the callbacks is created which aims to inform the federate for the new updates as soon as possible. With the help of the callbacks, the receiving federate detects that there are new object instances of its interest, and it can prepare to receive their attribute updates. The callbacks are handled in a separate thread to avoid the accumulative delay that the receiving process could produce if it would be included in the transmitting cycle. Now, the initialization process is finished, and the actual latency measurement set-up can start.

The core logic of the latency measurement set-ups remains unchanged: the transmitting federate updates the object instance attributes that it owns, and the receiving federate receives the updates and responds with an interaction. The time it takes for the update to get to the receiver added with the time for the response to get back to the transmitter form the round trip time (RTT) of the connection. The RTT cycle is presented in the Figure 11.

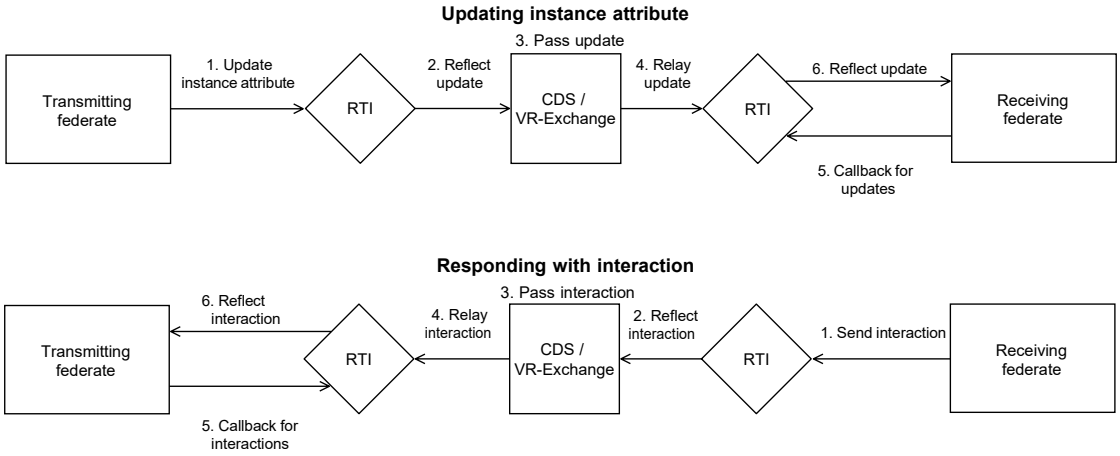

Figure 11: *The RTT cycle of the system.*

The overall latency consists of several parts as there are multiple steps in the RTT cycle. The transmitted message starts from the test bench software, which sends the attribute update to the RTI. Next, the RTI processes the update and relays it onwards and outside the first laptop. The update goes through the physical Ethernet cable and enters the second laptop containing the active CDS or VR-Exchange. Therefore, before any processing from the CDS, the message has travelled already through hardware and software, which add ups to the latency. After the CDS ruleset checking and further processing, the message reaches the final Ethernet cable and laptop, where the second RTI and the test bench software do their own procedures. Then, the interaction response from the receiving federate travels the same way but in the opposite direction. By the time the interaction is in the transmitting federate, the RTT cycle is complete.

When the RTT is divided by two, the end result is the latency of the system. By repeating this cycle of updates and responses 10000 times, summing the latencies together and dividing them by 20000, the comparable average latency of the system can be found. Because only the middle software in the configuration changes, CDS to VR-Exchange or vice-versa, the differences in latencies in the same measurement set-up should originate from the dissimilarities of the latencies of the CDS and VR-Exchange. The measuring steps of the latency set-up is described in the Algorithm 1.

The measuring process of average latency

Number of aircraft instances: `X = 1`,
number of iterations: `numberOfIterations = 10000`,
amount of time to wait: `sleepTime = 100 ms`,
at start: `i = 1, numberOfUpdates = 0 and numberOfResponses = 0.`

```
1. Save the startTime of the whole measurement
2. Federate A updates a single attribute of an aircraft instance
3. Save the updateTime of the update i
4. Federate B receives the update and numberOfUpdates ++
5. B sends a response (Collision interaction)
6. A receives the response and numberOfResponses ++ (concurrent
   Callback thread, already initialized before the start)
7. Save the responseTime of the response i
8.  if i <= numberOfIterations
9.     Wait for sleepTime, i++ and continue from row 2
10.  elseif i > 10000 AND numberOfResponses = 10000
11.    save EndTime
```

➔ 1) Calculate the differences of `responseTime – updateTime` for the latencies. Sum them together and divide by 20000 => **average latency.**
➔ 2) Plot the latency as a function of received responses.

Algorithm 1: *The measurement steps for the latency measurement.*

The same measuring process is then repeated by replacing the CDS with the VR-Exchange to get reference without the effect of the CDS. After the latencies are measured, the throughput set-up is executed.

The throughput measurements are performed with a similar test set-up, but now the messages are sent only in one way, from the transmitter to the receiver to find out the maximum throughput capability of the connecting element between the two RTIs. Now, as the messages go only in one way, it will reduces the complexity of the system and the initialization process. The initialization process is very similar to the latency measurement, but now some phases are unnecessary and can be removed. Therefore, the initialization process has fewer steps, and its simpler implementation is described in the Figure 12.

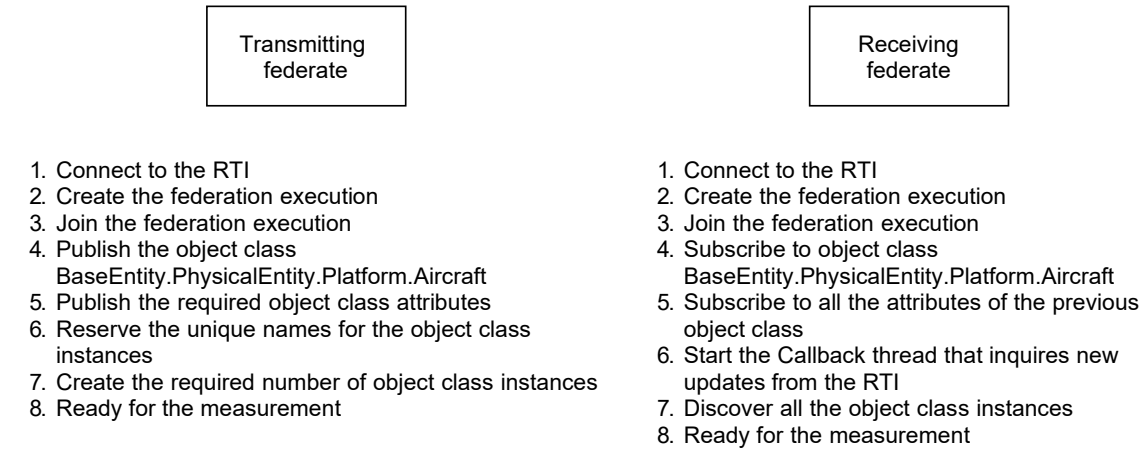

Figure 12: *The initialization process of the throughput measurement.*

In the throughput measurement, the federate A only transmits updates for the simultaneous aircraft instances while the federate B receives them. This measurement consists of 100000 updates, that are distributed evenly by the number of simultaneous updating aircraft entities and the number of iterations. For example, with 50 simultaneous entities, the number of iterations is 2000 to achieve a total of 100000 updates. There are different combinations of simultaneous entities and iterations to find out the optimal configuration for maximizing the throughput.

In the measurements with the CDS the delay parameter sleepTime is set to 0 ms. However, due to the message handling and processing of the VR-Exchange, the sleepTime is not 0 ms. The sleepTime adds some delay between the iterations in order to prevent the interruptions of the flow of the messages. It was discovered that the transfer of the entity updates slows down remarkably, and some updates are lost if there are too many updates to process in a short time interval. But by keeping the delay time as low as possible and calculating the throughput with the actual number of received entity updates and the total time taken, the maximum throughput can be calculated for the CDS and VR-Exchange. The detailed description of the measurement steps of the throughput measurement can be found from the Algorithm 2.

The measuring process of maximum throughput

Number of aircraft instances: `X = 50/100/150/200/250,`
number of iterations: `numberOfIterations = 2000/1000/666/500/400,`
amount of time to wait: `sleepTime = 0/30/40/50/60/70/80/90/100 ms,`
at start: `i = 1 and numberOfReceivedUpdates = 0.`

```
1. Save the startTime of the whole measurement
2. Federate A updates the attributes of all aircraft instances X
3. Federate B receives the updates, numberOfReceivedUpdates ++
4. Save the receiveTime of the updates.
5. if i < numberOfIterations
6.      Wait for sleepTime, i++ and continue from row 2
7. else
8.    save endTime and numberOfReceivedUpdates
```

➔ 1) Calculate (`numberOfReceivedUpdates)·1000/(endTime-startTime)` => **throughput**

Algorithm 2: *The measurement steps for the throughput measurement.*

Both measurements produce a result log-file that contains the values of interest in a readable comma-separated values (CSV) form. The CSV-file for the latency measurement contains the index of the attribute update, the timestamps when the update is sent and the interaction is received, and the calculated latency. The throughput equivalent has the index of the received update, timestamp for the time it is received and the start and end times of the measurement. It does not have the transmission time for the entity update because of the differing clock times of the laptop 1 and laptop 2. The Algorithm 1 guarantees that all the transmitted messages get a response. However, in the Algorithm 2, it is possible that some of the transmitted messages are not received. But this is acknowledged in the final throughput calculation which is calculated with the number of updates that are properly received in the receiving federate. The end results of all the measurements are presented in the next chapter with the help of graphical charts and tables.

6. PERFORMANCE RESULTS

The performance results are divided into two subsections: the first one presents the results for the latency measurements where two tables and two graphs are introduced. The second subsection summarizes the throughput measurement. Within both these measurement set-ups, the core measurement procedures do not change between the CDS and its reference the VR-Exchange.

6.1 The latency set-up

The latency measurement contained 10000 samples for both the measurement with CDS and VR-Exchange, which were extracted from the CSV-log file after the measurements. The values presented in the Table 4 were calculated with spreadsheet program Microsoft Excel. The key figures that were calculated were the average, the standard deviation, the minimum and maximum.

Table 4: *Latency set-up result table.*

Software	Average (ms)	Std (ms)	Min (ms)	Max (ms)	Samples
VR-Exchange	13.7	7.2	2.5	55.5	10000
CDS	3.9	1.6	2.0	44.5	10000

To present the development of the latency and to find out if there are any major irregularities as the simulation progresses, a graph for the latency results was also made. The same data was exported to MATLAB, which allowed its further processing. The latency as a function of received responses is presented in the Figure 13.

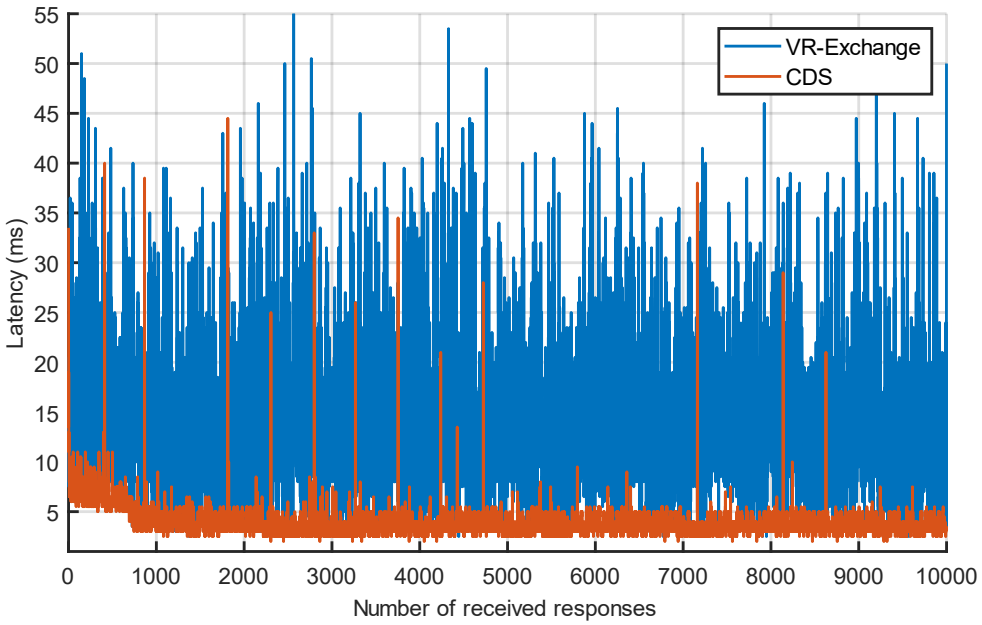

Figure 13: *Latency as a function of received responses.*

In addition, MATLAB allowed to gather more information about the distribution of the data of the latency measurement. The cumulative distribution function (CDF) of the measurement is plotted in the Figure 14. The graph shows the latency and the corresponding percentage of values of the whole population that are below the desired latency value.

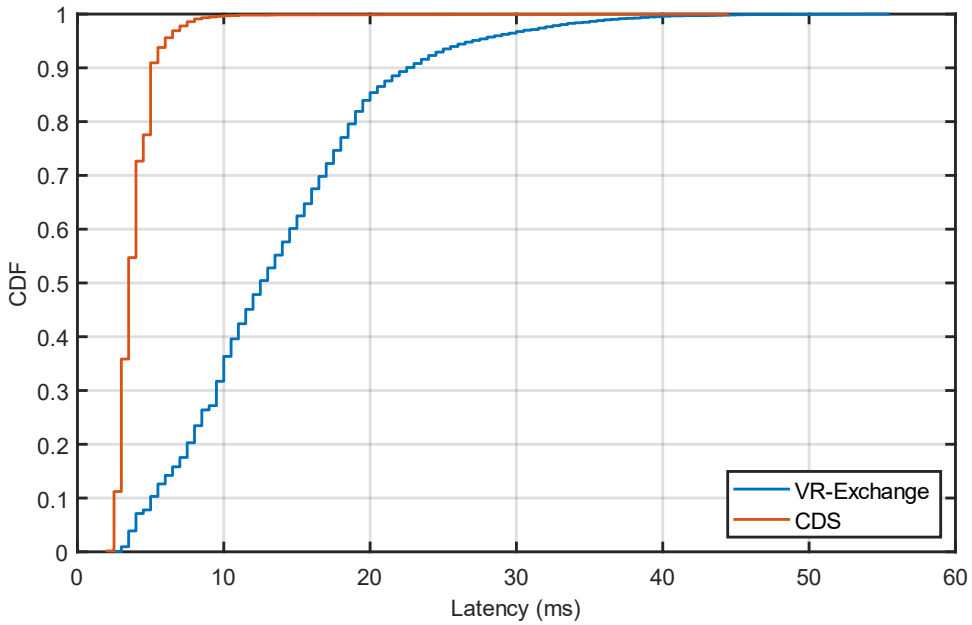

Figure 14: *Cumulative distribution function (CDF) of the latency results.*

Some of the most important percentiles are also highlighted in the Table 5, where the 5th, 50th and the 95th percentiles are presented for the CDS and VR-Exchange. Other percentiles, including the ones presented in the Table 5, can be roughly approximated from the CDF plot in the Figure 14.

Table 5: *Latency set-up percentiles table.*

Software	5th percentile (ms)	Median (ms)	95th percentile (ms)	Samples
VR-Exchange	4	12.5	27	10000
CDS	2.5	3.5	6	10000

6.2 The throughput set-up

The throughput measurement consisted of 10 set-ups where the total number of transmitted updates was determined by multiplying the number of simultaneous aircraft instances with the number of iterations. The preferred sleep times of the VR-Exchange set-ups were determined by testing different values and repeating the measurement. The optimal delay value was chosen when it resulted in the highest throughput before the interruptions of the flow of the messages occurred. The results for the throughput measurement can be seen from the Table 6.

Table 6: *Throughput set-up result table.*

Software	Number of instances	Number of iterations	Sleep time (ms)	Time (ms)	Received updates	Throughput (entities/s)
VR-E	50	2000	30	94324	78059	827.6
CDS	50	2000	0	277875	100000	359.9
VR-E	100	1000	40	66183	62719	947.7
CDS	100	1000	0	268286	100000	372.7
VR-E	150	666	60	79249	74607	941.4
CDS	150	666	0	267057	99900	374.1
VR-E	200	500	70	70768	73543	1039.2
CDS	200	500	0	258803	100000	386.4
VR-E	250	400	80	64048	68977	1077.0
CDS	250	400	0	266411	100000	375.4

In the following chapter, the meaning of the achieved results is discussed in more detail. The results are also compared to the performance requirements, which were defined earlier, to find out if the CDS fulfills these requirements.

7. CONCLUSIONS

The two main purposes of this thesis were to find out suitable performance requirements for a real-time simulation network and to evaluate the capabilities of a corresponding real-time simulation implementation using CDS. The proper performance requirements were established in the forms of latency and throughput due to the needs of small delays, but also high capacity demands of the real-time simulation networks. It was discovered that the proper upper limit for the latency would be 100 ms whereas the throughput requirement would be 1870 entities/s.

Moreover, these performance requirements were applied to the measurements with the CDS which measured the latency of a single updating entity and the maximum throughput with multiple simultaneously updating entities. The results showed that the average latency of a single updating entity remain below 100 ms when the CDS connects the networks. In contrast, the maximum throughput of the system with CDS was 386.4 entities/s, which was below the requirement of 1870 entities/s. Thus, these results indicate that the CDS achieves the latency requirement if there are not too many simultaneous updating entities but fails to meet the throughput requirement.

For the comparison, same measurements were made by changing the network connecting element from the CDS to the VR-Exchange. The VR-Exchange should only monitor the data going through it, so any differences with the end results between the CDS and the VR-Exchange should originate from the additional processing of the CDS. The VR-Exchange achieved a maximum throughput of 1077 entities/s, which was considerably larger than the corresponding result of the CDS. Surprisingly, the average and the standard deviation of the single entity latency measurement for the VR-Exchange, were significantly greater than the results of the CDS. Therefore, the accurate description of how much latency does the CDS add to the system cannot be made in the basis of the single entity measurement. It would be beneficial to perform similar latency measurements where there are more than one updating entity at the same time in order to have more information of the behavior of the latency in different scenarios. Also, additional reference measurements could be made, where the VR-Exchange could be replaced with another similar software if there are other HLA supporting software available.

The reason for the unexpected behavior of the VR-Exchange with a single updating entity remains unclear, but there are also other aspects that could change the end results. For

example, the utilization of more realistic types of attribute updates that are sent within the federates. Now, the transmitting federate updates the aircraft entity by sending four attributes: EntityIdentifier, EntityType, Spatial and ForceIdentifier. The first three are mandatory for the message to go through the VR-Exchange, and the last parameter ForceIdentifier is updated in every iteration to receive updated data. However, in a real-life scenario, there would be more possible attribute values, and each of them can have an alternating update rate. For example, the spatial attribute which contains the information regarding the location, position, and motion of the entity, can update very frequently if the entity makes fast changes in its course. When the position and directions of the entity remains more constant, some of the parameters are unchanged and the updates are not sent as frequently. The high standard deviation of the HLA messages was also present in the commercial flight simulator measurement presented in Table 3. The results of this thesis might not be as accurate in some use cases, which differ remarkably from the basic set-ups of the measurement cases. Thus, repeating the same experiments with recorded data from the real cases or even by connecting the CDS to a live and operational network would be an interesting topic for validation and further research and development.

Similarly, the throughput requirement of 1870 entities/s is an indicative value which was established from the basis of one commercial flight simulator and from the weighted average of the opinions of different experts. Thus, it is not the absolute truth which determines if a performance of a real-time simulator system is enough, and the final requirement can change if the weighting of the opinions or the flight simulator are changed. However, the difference between the measured maximum throughput of CDS at 386.4 entities/s is clearly smaller than the requirement, so improvements and optimizations to the throughput capabilities of the CDS should be made to meet the requirements better.

In addition, it seems that the relaying capabilities of the RTIs do change as the simulation progresses. For example, in the Figure 13 the average latency drops remarkably with both CDS and VR-Exchange, when 800 iterations have passed. This would indicate that it takes less time for the RTIs to process the messages when the simulation progresses to a certain point, which would mean better performance for the network. To find out if this is a RTI manufacturer specific feature, other RTI-implementations could be tested with the same test measurement set-ups. Also, it would give more information and the results of this thesis could be analyzed more if similar measurements would be measured with these other RTI-implementations. There are several alternatives among different RTI contributors: Pitch Technologies is a commercial solution, like the RTIs of the MAK

Technologies used in the measurements, whereas Portico and CERTI are similar open-source alternatives with potential to grow and perform as their commercial counterparts [26][27]. Also, the possible interoperability between the different manufacturer's RTIs and the effect of the required message conversions to the performance of the network should be studied more. By supporting the utilization of various RTIs together from different RTI-manufacturers, the improved diversity of the network would allow even more different systems and operators to join the networks while also reducing the dependency to a single commercial component manufacturer.

As a conclusion, the CDS achieves the latency requirement, but only with single updating entity. However, the CDS would need some improvement in its software to meet the throughput requirement. If the performance of the software were improved, for example by utilizing more concurrent threads in suitable locations and by spotting and cleaning up inefficient methods, both performance requirements could possibly be achieved and exceeded.

REFERENCES

[1] Topçu O., Oğuztüzün H., Guide to Distributed Simulation with HLA, Cham: Springer International Publishing, 2017, 307 p.

[2] Tolk A., Engineering principles of combat modeling and distributed simulation, New York: WILEY, 2012, 888 p.

[3] Banks J., Introduction to Simulation, WSC'99 Winter Simulation Conference Proceedings. 'Simulation - A Bridge to the Future', 1999, pp. 7–13

[4] Funke J., Computer-based Testing and Training with Scenarios from Complex Problem-solving Research: Advantages and Disadvantages, International journal of selection and assessment, 1998, pp. 90–96.

[5] IEEE, The Standards Development Lifecycle, 2021, Available (accessed on 24.03.2021): https://standards.ieee.org/develop/index.html

[6] IEEE, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules, 2010, 26 p.

[7] Dahmann J., Morse M., "High Level Architecture for Simulation: An Update", Proceedings. 2nd International Workshop on Distributed Interactive Simulation and Real-Time Applications, 1998, pp. 32–40

[8] Straßburger S., Overview about the High Level Architecture for Modelling and Simulation and Recent Developments. Simulation News Europe 16, 2006, pp. 5–14

[9] IEEE, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Object Model Template (OMT) Specification, 2010, 100 p.

[10] Möller B., Karlsson M., Herzog R., Wood D., Security in Simulation – New Authorization Opportunities in HLA 4, Virtual Simulation Innovation Workshop, 2021

[11] Möller B., Dubois A., Leydour P., Verhage R., RPR FOM 2.0: A Federation Object Model for Defense Simulations, Fall Simulation Interoperability Workshop, 2014

[12] IEEE, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Federate Interface Specification Specification, 2010, 363 p.

[13] Pu I.M., Fundamental data compression, Oxford: Butterworth-Heinemann, 2006, 269 p.

[14] ISO/IEC, ISO/IEC 21320-1:2015 Information technology – Document Container File – Part 1: Core, 2015, 8 p.

[15] Wallace G., The JPEG Still Picture Compression Standard, Communications of the ACM 34.4, 1991, pp. 30–44

[16] Shinde S., Computer Network. Daryaganj, New Age International Ltd, 2000, 404 p.

[17] Holmala O., Designing a Protocol Agnostic Rule Engine for a Cross-Domain Solution, Tampere University, 2019, Available: http://urn.fi/URN:NBN:fi:tty-201905031471

[18] Gervais C, Chaudron J.B., Siron P., Leconte R., Saussie D., Real-Time Distributed Aircraft Simulation through HLA, Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications (DS-RT '12), IEEE Computer Society, 2012

[19] Nandi S., Rodriguez F., Wood D., Granowetter L., The MAK High-Performance RTI : Performance by Design, 2015, Available: https://edstechnologies.com/Mailer/may15/newsletter/images/MAK_RTI_Performance.pdf

[20] Ping I., HLA performance measurement, Naval Postgraduate School, 2000, 105 p., Available: https://archive.org/details/hlaperformanceme109457709/page/n9/mode/2up?q=10

[21] Lockheed Martin, Prepar3D professional plus, 2020, Available (accessed on 05.04.2021): https://www.prepar3d.com

[22] Wireshark Foundation, Wireshark, 2021, Available (accessed on 05.04.2021): https://www.wireshark.org/

[23] MAK technologies, VR-Exchange, 2021, Available (accessed on 05.04.2021): https://www.mak.com/products/link/vr-exchange

[24] Oracle, Javax.swing package, 2020, Available (accessed on 01.05.2021): https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html#package_description

[25] Experts from Insta DefSec Oy, Insta DefSec Oy, Tampere, Interview on 30.04.2021.

[26] Gütlein M., Baron W., Renner C., Djanatliev A., Performance Evaluation of HLA RTI Implementations, 2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2020, pp. 1–8

[27] Akram A., Sarfraz M. S., Shoaib U., HLA Run Time Infrastructure: A Comparative Study, Mehran University Research Journal of Engineering and Technology, 2019, pp. 961–972