Enhancing Load Balancing Efficiency Based on Migration Delay for Distributed Virtual Simulations

By

Turki Alghamdi

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements for the degree of

Master of Applied Science for Computer Engineering

In Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering

Abstract

Load management is an essential and important factor for distributed simulations running on shared resources due to load imbalances that can caused considerable performance loss. High Level Architecture (HLA) -based simulation is a framework that works to facilitate the design and management of distributed simulations. HLA coordinates the interaction between simulation entities (federates). However, HLA-based simulation standards do not present the ability to manage resources or help detect load imbalances that could directly cause decrease of performance. Focusing on this constraint, a migration-aware dynamic balancing system has been designed for HLA simulations to offer an efficient load-balancing scheme that works in large-scale environments. This system presents some limitations on estimating costs and benefits, so we propose an enhancement to this existing load balancing system, which improves the accuracy of estimating the number of migrations for the next load redistribution. The proposed scheme detects the load imbalances by evaluating the recourses overhead. The scheme classifies the recourses based on the overhead as overloaded and underloaded, followed by matching the highest overloaded recourses with the lowest underloaded recourses. Furthermore, the proposed scheme aims to precisely estimate the number of migrations by evaluating and analyzing the recourses to obtain the best number of migrations. Therefore, certain migrations that do not contribute to an improvement in the simulation performance are avoided. This avoidance is based on comparing time delay and time gain. Moreover, to be considered for migration, the overall sum of the time gains should be larger than the overall sum of the time delays. The proposed scheme has shown an improvement on decreasing the execution time.

Acknowledgment

First, with gratitude I thank Allah for giving me the strength and patience to finish an important chapter of my life. Secondly, I would like to express my deeply felt gratitude to my supervisor, Professor Azzedine Boukerche of the School of Electrical Engineering and Computer Science (EECS) at The University of Ottawa for his diligent guidance and supervision, his advices, encouragement, enduring patience, and constant support. Without him this thesis would never have been finished. Words cannot begin to express my appreciation and gratitude towards Dr. Robson Eduardo De Grande who has shown great help in the completion of this thesis through his guidance and his gentle and kind approach. I would also like to thank my friends and col- leagues who I have met during the process of earning my Master's at the PARADISE re- search laboratory for always being helpful and optimistic, especially Amar Farouk Merah. I wish them all the best in their studies and careers. Moreover, I would like to thank my parents, Jomah and Gormollah Alghamdi, who always believed in me. They taught and encouraged me to always be myself. All thanks and gratitude go to my siblings: Muhammad, Salehah, Abdullah, Azza, Zohor, and Najat. I hope they will all have the chance to achieve their best in their current and future endeavors. Many thanks are due to Al Jouf University, Saudi Arabia, through which my dream of pursuing my Master's degree in Canada came true.

Last but definitely not least, I offer my thanks to those people who are very important in my life. They have pushed me really hard to finish my work. I thank my lovely wife, Nawal, for helping me and for always providing a suitable environment for me to finish my work. I thank my daughters, Weaam, Wateen, and the new baby (Rateel) who we are expecting to see very soon. You wanted me to help you with your homework but a lot of days I could not because of my work. I appreciate your patience.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface
CCM	Contractual Computing Mechanism
CLB	Cluster Load Balancer
CLM	Computation Load Monitor
DDM	Data Distribution Management
DDMS	Data Distribution Management Service
DMS	Declaration Management Service
FedExec	Federation Executive process
FMS	Federate Management Service
FOM	Federation Object Model
FT	Fault-Tolerant
FTP	File Transfer Protocol
FT-RSS	Fault-Tolerant Resource Sharing System
GA	GA Grid Agent
GCE	Grid Computing Environment
GDSA	Grid-based Distributed Simulation Architecture
GGF	Global Grid Forum
G-HLAM	Grid HLA Management System
GPDS	Grid-Based Parallel and Distributed Simulation environment
GRAM	Grid Resource Allocation Manager

GridFTP	Reliable Data Transfer
GVT	Global Virtual Time
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
IPC	Inter-Processor Communication
libRTI	Run-Time Infrastructure library
LLB	Local Load Balancer
LMI	Local Migration Interface
LMS	Load Management System
LP	Logical Process
LRC	Local Run-Time Infrastructure Component
LS	Local Service
LVT	Least Virtual Time
MDS	Monitoring and Discovery Service
MIS	Monitoring Information Service
MM	Migration Manager
MOM	Management Object Model
OGSA	Open Grid Services Architecture
OMT	Object Model Template
OOMS	Object and Ownership Management Service
PAT	Processor Advance Time
PDES	Parallel Discrete Event Simulation
PSDE	Parallel and Distributed Simulation Environment

QAP	Quadratic Assignment Problem
QoS	Quality of Service
RSS	Resource Sharing System
RTI	Run-Time Infrastructure
RTIExec	Run-Time Infrastructure Executive process
SA	Simulation Agent
SimKernel	Simulation Kernel
SOHR	Service Oriented HLA RTI
SOM	Simulation Object Model
TMS	Time Management Service
VTP	Virtual Time Progress

Chapter 1. Introduction

The dynamic management of load in large-scale distributed simulations, such as High Level Architecture (HLA)-based distributed simulations, has been implemented in several complex areas. These HLA-based simulations have been used to help researchers to identify situational problems and focus on solving these issues. Several applications have employed HLA-based simulations such as wireless and mobile networks which are used to test new communication protocols [1] [2]; or flight simulations which are used to check the influence of atmosphere on flight [3], test new air traffic management techniques [4], or observe combat between aircrafts [5].

The most important aspect of a distributed simulation, particularly a large-scale one, is its performance; however, many issues affect performance. For example, load imbalances cause an irregular arrangement of simulation load during run time. Moreover, uneven partitioning causes some resources to be overloaded while other resources remain underloaded. Other issues include improper distribution of the simulation between resources and resource heterogeneity.

A static balancing method provides a load management system for the distributed simulations. This is based on resource heterogeneity and can avoid load imbalances. However, dynamic load-changing cannot be solved by using the static method. Thus, a dynamic load-balancing system has been devised in order to detect load imbalances by monitoring and observing load changes. The dynamic balancing of computation and communication load is employed to manage the shared resources more efficiently.

High Level Architecture (HLA) is a framework that works to facilitate the design and management of distributed simulations. HLA manages the interactions between simulation entities (federates). Furthermore, HLA allows the reuse of federates in different distributed simulations, which provides a greater capability of communication between federates. Therefore, HLA standards are divided into rules and services. HLA rules must be followed by simulation entities (federates) and distributed simulations (federations). Grid services [6] improve the simulation performance. For example, Interface Specification is defined by Run-Time Infrastructure (RTI) services. The Object Model Template (OMT) is responsible for recording information by providing a common method and by establishing the format of the following key models: Federation Object Model (FOM), Simulation Object Model (SOM), and Management Object Model (MOM) [7] [8] [9].

However, HLA standards have certain limitations, such as not providing any load balancing technique, not having an ability to control the distributed simulation entities on shared resources, and not providing a federate migration protocol that moves the federate without freezing the whole distributed simulation.

1.1. Thesis Statement

As mentioned above, HLA standards have limitations. These limitations cause performance loss, which results in jeopardizing the simulation system. The load balancing system is a critical issue which allows better performance of the simulation system; thus, the balancing system must reduce the execution simulation time. Some existing systems suggest solutions for this issue, such as centralized and dynamic schemes [10]. Centralized

and dynamic schemes aim to observe and control the load balancing capabilities of HLA-based simulation in large-scale environments, such as by measuring and analyzing the past migration to estimate a migration delay. However, the most important role of HLA-based simulation is to decrease execution time through even re-distribution of load, which results in improved simulation performance.

Load balancing schemes are divided into computational [11] [12] and communication [13] loads. In a number of non-dedicated resources, computational load balancing issues have been solved by distribution in multiple knapsacks [14], which have led to achievement of the optimal solution. Communication load balancing is based on the interaction load, which reduces the Quadratic Assignment Problem (QAP). This leads to another NP-hard problem. Thus, heuristics have been used to provide reasonable sub-optimal solutions that might achieve performance gain.

1.2. Motivation

HLA-based simulation standards do not have the ability to manage resources or help detect load imbalances that could directly cause decreased performance of distributed simulations. Consequently, optimistic and conservative simulations are needed to control and organize load distribution. Also, Grid Services [6] are needed to facilitate the monitoring of distributed resources and manage the resource system.

By reacting to load imbalances while a simulation is running, balancing schemes have used heuristics to distribute the load in a reasonable time. Analysis of heuristics provides a possible migration route, which improves the balancing system environment.

However, systems that are based on this work have used heuristics of past migration time and distance of past migration [15] [16]. So a brief description is necessary to understand how the system functions, the previous system uses third party tools such as Grid Services to determine CPU consumption, Ganglia, and HawKEye to monitor the balancing system. Furthermore, the balancing system uses CPU consumption as a metric to identify the federate status (overloaded and underloaded). Furthermore, the estimation of migration delay is based on past migration time, distance of past migration, and distance of estimated migration. Time gain is based on current load of source resource, estimated minimum load, and current load of destination resource. The balancing system estimates the migration delay and time gain and applies a comparison between them. Thus, the balancing system measures and analyzes this information to make federate migration decisions. Therefore, the limitations on the decision-making for generating migration moves generated a research work to improve estimations and calculations to determine the most beneficial migration situation in light of performance gain according to time gain and migration costs.

1.3. Objectives

As the work of this thesis is incremental but independent of the previous dynamic load-balancing scheme, this balancing system is employed in order to enhance the estimation accuracy of time delay and time gain to re-distribute the load more precisely; thus, a number of variants are proposed. The proposed solutions are divided into two parts. The first part changes the estimation delay and estimation of time gain metric. However, the

estimation is based on the heuristics information while the proposed solution is based on current status or information along with the previous process time. Furthermore, a tool called *nload* is employed to monitor the current bandwidth usage. The second part increases the flexibility of Δt, which has been a problem in previous work due to having a fixed variable. Furthermore, the flexibility of Δt is based on the heuristics and time information of past migrations.

1.4. Contributions

As we have mentioned before, our proposed solution for the estimation of migration delay and time gain brings an enhancement to re-distribute load more accurately. The proposed techniques are developed by changing the estimation delay and estimation of time gain metrics. Besides that, we added *nload* tool into the monitoring tools to provide the current bandwidth usage which allows to evaluate the transmutation availability. The estimation of Δt has been also modified to be flexibly adapted by using the past migration time as heuristics information, which shows ability to avoid the costly migrations. The proposed solution thus improved the estimation of the migration delay to be more precise and thereby improve the performance and re-distribution of the load.

1.5. Thesis Outline

This thesis is organized as follows. Chapter 2 introduces background information and presents HLA standards as well as HLA limitations. Also, this chapter outlines the main

characteristics of Grid Services. Next, Chapter 3 presents the related work, including existing dynamic load redistribution solutions and drawbacks. Chapter 4 describes in detail our proposed solution of enhancing load balancing efficiency based on migration delay. Chapter 5 explains the experimental details and results. Finally, Chapter 6 offers the conclusion and discusses future work.

Chapter 2. Background Information

This chapter presents important information related to High Level Architecture (HLA) and reviews the existing research on this topic. This chapter will identify the motivation, purpose, and challenging issues of large-scale HLA-based simulation, including certain limitations that were encountered while the HLA-based simulation was running. Furthermore, we will also discuss other aspects of the topic, such as dynamic load balancing. More information about load balancing is presented in Chapter 3.

2.1. High Level Architecture

High Level Architecture (HLA) has been developed to provide a general-purpose standard method for designing and coordinating distributed simulations. This standard, which allows for the execution of distributed, parallel simulations for military purposes, and was developed by the United States Department of Defense. In 2000, the HLA specification became an open IEEE standard [9]. Also, HLA became a recommended procedure for developing interoperable parallel simulations.

HLA was introduced to provide interoperability and to avoid causality discrepancies and reusability in simulation design and management mechanisms. Federation (HLA-based simulation) is designed to interact with federates (independent components). Therefore, federation avoids causality discrepancies by interacting with another

federation through a defined communication method. Subsequently, execution of the federation is an actual run-time simulation execution.

Three components in HLA apply the rules: Interface Specification, Object Model Template (OMT), and Rules. Interface Specification is defined by Run-Time Infrastructure (RTI) services and is responsible for determining the "callback" functions of each federate [17] [18]. OMT is responsible for recording information by providing a common method and by establishing the format of the following key models: Federation Object Model (FOM), Simulation Object Model (SOM), and Management Object Model (MOM). The rules ensure the proper interaction of simulation in a federation and describe the simulation and federate responsibilities. Thus, there is no difference between HLA-based simulation and any other type of modeling and simulation application because the HLA-based simulation provides the Federation formalism which allow the federates to be modeled such that the framework can support Federation Execution.

The RTI is software that provides common services to the simulation system. These RTI services are classified into six management areas: Federation Management, Declaration Management, Object Management, Ownership Management, Time Management, and Data Distribution Management (DDM). These management services work by filtering [19]and determining the simulation interaction between federates with a limitation that avoids the issues of simulation modeling.

As shown in Figure 2.1, the RTI middleware consists of the following: an RTI Execution process (RTIExec), a Federation Execution process (FedExec), and the RTI library (libRTI). The RTIExec is responsible for coordinating between Federation and FedExec. Therefore, the FedExec manages the simulation of federation. The libRTI

works to make HLA service methods available to federates and manages the distributed simulation. The libRTI has the mechanisms that can provide the communication between RTIExec, FedExce, and other federates.

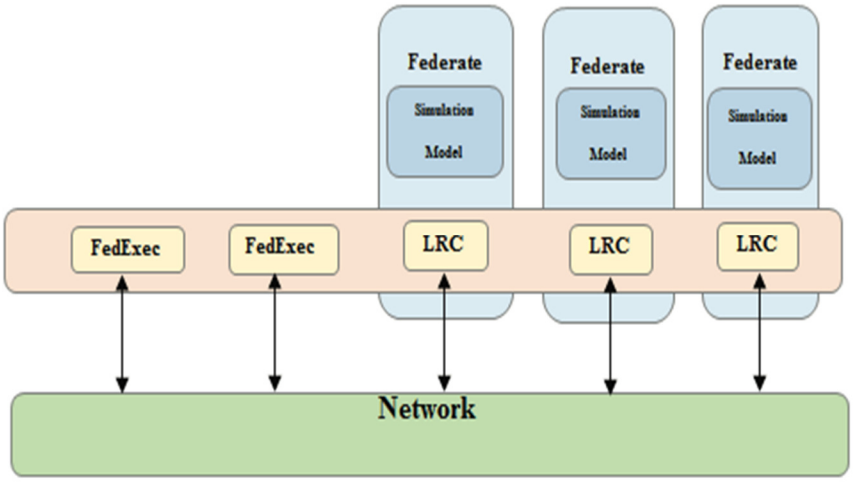


Figure 2.1 HLA Architecture *[20]*

The HLA simulation with management services allows federates to coordinate data exchange with every operation. The RTI observes the communication between federates. The Local RTI (LRC) allows federates to access the libRTI interface through LCR.

Data Distribution Management (DDM), which is one of the management services, works with HLA to attempt to minimize the communication overhead. However, the HLA standard does not offer any solution for balancing the computational load and communication rate. Therefore, DDM allows for data exchange actions between federates, such as publish and subscribe actions. DDM provides management techniques to observe the transmission between federates and to restrict the communication to relevant data. DDM allows higher available bandwidth to decrease any network overhead that is caused by simulation.

The HLA standard does not provide any technique or method to control the distributed simulation load that is placed on shared resources when the simulation is running. HLA provides a mechanism to ensure reliability, show the communication capacity for simulation applications, and show enough computing power. To avoid jeopardizing the HLA simulation caused by overloaded resources, load balancing is required. The load balancing attempts to maximize resource utilization and minimize communication delay in order to improve the performance of HLA simulation.

2.2. Grid Computing

As we know, Grid computing is accessed by the shared resources that provide various services [21]. The authors in [22] have proposed a grid system that can organize the distributed application that runs in shared resources. The aim of grid computing is to have a flexible and secure system. Grid computing provides a mechanism that can deduct non-interactive processes and exchange data even with a large amount of files. This grid computing presents a super virtual computer view of the end user by managing the workload in the resources. Grid computing works with large and complex tasks in a computing server [23].

According to Foster [6], an Open Grid Services Architecture (OGSA) has been-proposed to work with a Global Grid Forum (GGF) that provides an architecture for a service-oriented grid computing environment. The GGF is an application system that is employed in scientific and business applications. The OGSA grid specifications are classified into the following: infrastructure services, self-management services, information

services, data services, resource management services, execution management services, and security services. In fact, the grid computing services observe distributed application and scheduling and allocation resources, and can meet all requirements of applications [24].

Indeed, the OGSA of many grid projects has been devised into a Globus Toolkit [25]. In fact, the Globus Toolkit has become a middleware standard for Grid computing. The Globus Toolkit has basic tools that can monitor the balancing system. However, the Globus Toolkit has not yet solved the issues that come with load imbalances. Therefore, the balancing system in the Globus Toolkit allows for the load of computing resources to be retrieved and for the load transfer to be programmed.

2.3. Large-Scale based on HLA-based Simulations

As we know, Data Distribution Management (DDM), which is one of the management services, works with HLA to attempt to minimize communication overhead [26]. However, The HLA standard does not present any solution for balancing the computational load and communication rate.

According to S. Zhu, the HLA has offered little support to large-scale distributed simulation [27].

The HLA standard does not provide any technique or method to control the distributed simulation load that is placed on shared resources when the simulation is running. The HLA simulation can jeopardize the entire simulation that is caused by overloaded resources or local failure. However, management systems are required to have a satiable

distributed simulation without overloaded resources causing even a single local failure. According to [28], a faultless HLA-based simulation is assumed. In addition, other authors [29] [30] assume that the HLA has not included any formal failure model that detects any issues regarding fault-tolerance. According to K. Zajac, HLA has not provided any mechanism or tools that can manage the execution of the simulation on distributed resources [31]. According to the author in [32], the HLA standard does not provide a mechanism that can support federate migration, which is essential for dynamic setup on a Grid Application. According to [33], the OGSA supports HLA-based simulation, and is therefore suitable for HLA-based simulation. However, the OGSA is useful for fault-tolerance in the distributed simulation.

Evidently, managing the large-scale HLA simulation is enabled by employing the capability of grid services.

2.3.1 Resource Sharing System

The Resource Sharing System (RSS) and the Fault-Tolerant Resource Sharing System (FT-RSS) works as a manager for HLA-based simulation in large-scale environments. The manager is composed of two parts: monitoring of simulation load and detection of faults during run-time.

2.3.1.1 Load Balancing in the Resource Sharing System

The authors in [29] proposed a framework of FT-RSS that can manage the load for each node during the execution of a distributed HLA-based simulation system, as shown in Figure 2.2. The RSS consists of a manager, clients, and a communication federate. The manager is responsible for the node's load during the operation of the distributed load

balancing system. The clients work to observe and manage the simulation federates. The Communication Federate is responsible for communicating between the simulation federate and the manager. The RTI is responsible for managing the communication between the communication federate and the simulation federate, so when the balancing system decides to migrate federates from a node that gives a bad adaption effort to another node that shows a reasonable effort, it will save their states in a file. Then the new node will restart using the saved file.

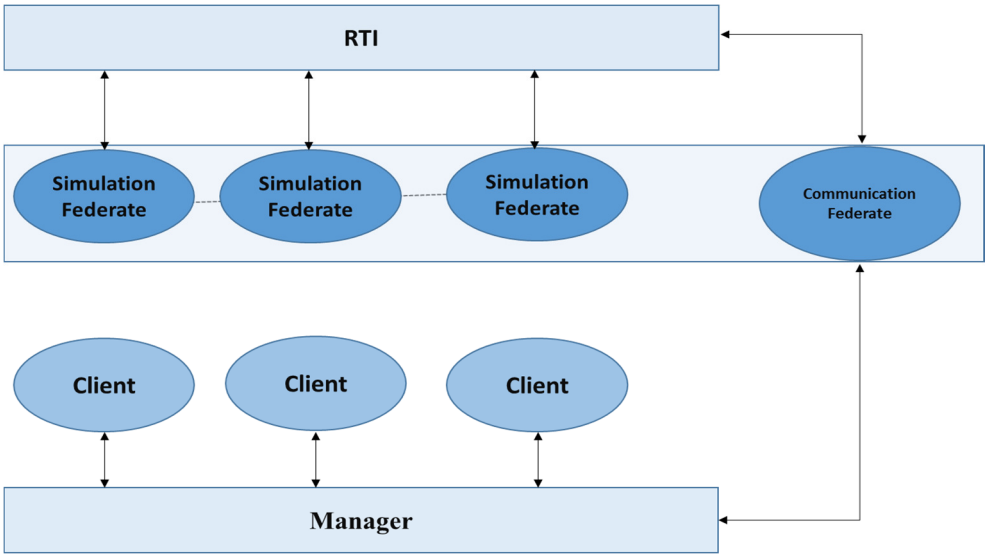


Figure 2.2 RSS Architecture *[20]*

2.3.1.2 Fault Tolerances in the Resource Sharing System

The author in [29] proposed an FT tool in FT-RSS that can allow configuring of fault-tolerance mechanisms. However, the FT-RSS framework that has been inserted into the FT tools monitors and observes the errors and failures. The FT tools are responsible for detecting the errors and providing the error mechanism for federates.

2.3.2Load Management System

The author of [34] developed a Load Management System (LMS) that supports HLA-based distributed simulation of geographically different organizations with efficient and effective simulations.

A Globus Toolkit [35] [36] was devised to enable grid computing, as shown in Figure 2.3. The LMS uses the Globus toolkit to manage all resources in a distributed simulation. Therefore, the RTI supports and manages the execution of the simulation federation.

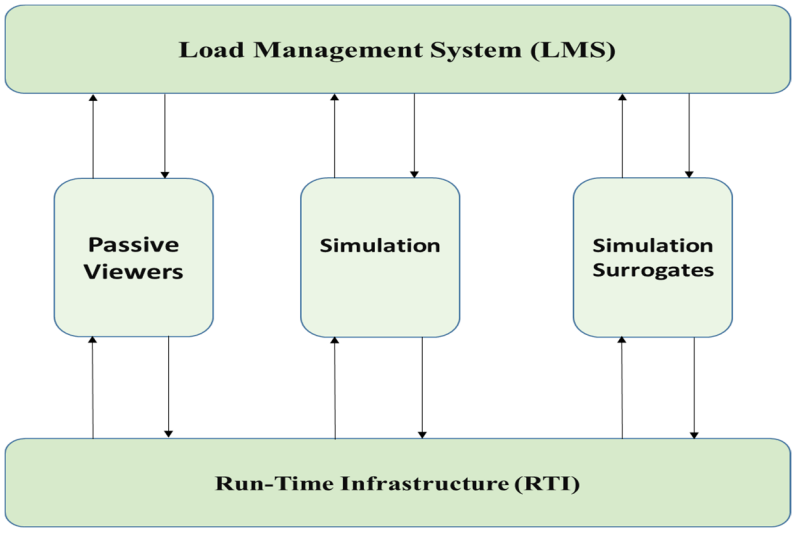


Figure 2.3 Load Management System Architecture with HLA based simulation *[20]*

2.3.3 Simulation Kernel (SimKernel)

The SimKernel is a framework that has been developed by [37] to simplify the simulation design by using RTI to execute the parallel simulation. The SimKernel framework is divided into the following components: a federate execution model, an automatic code generation tool, and a new layer that has been placed between the user and the HLA-based

simulation. As shown in Figure 2.4, the framework consists of Logic Processors (LPs) that are mapped as federates and messages that are mapped as RTI interaction. In addition, the SimKernel uses DDM to limit the number of messages that need to be transmitted.

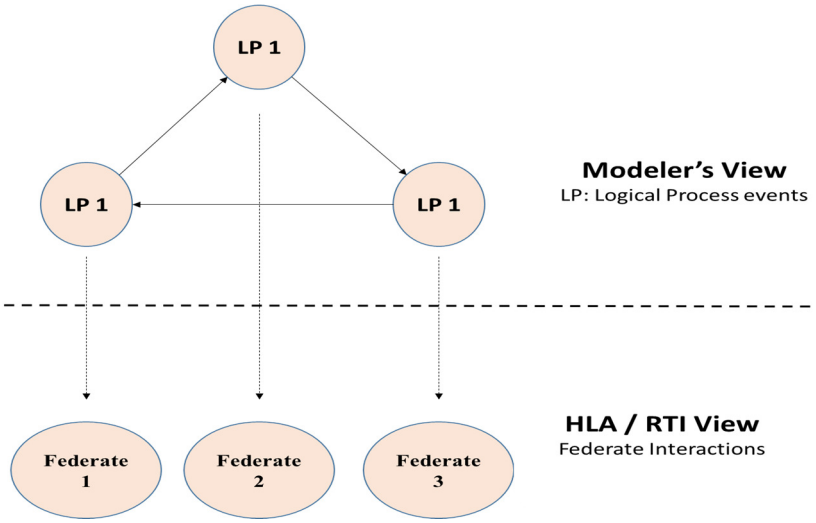


Figure 2.4 LP with RTI Federate Mapping *[20]*

2.3.4Grid HLA Management System

The authors of [38], [39], [40], and [41] have proposed a Grid HLA Management System (G-HLAM). The G-HLAM has been developed to manage the execution based on HLA simulation using a grid environment. This was designed based on the OGSA concept. Therefore, this framework supports federate migration, which improves the simulation performance. As shown in Figure 2.5 [36], G-HLAM framework is composed of three services: a Broker Service, a Performance Service, and a Registry Service.

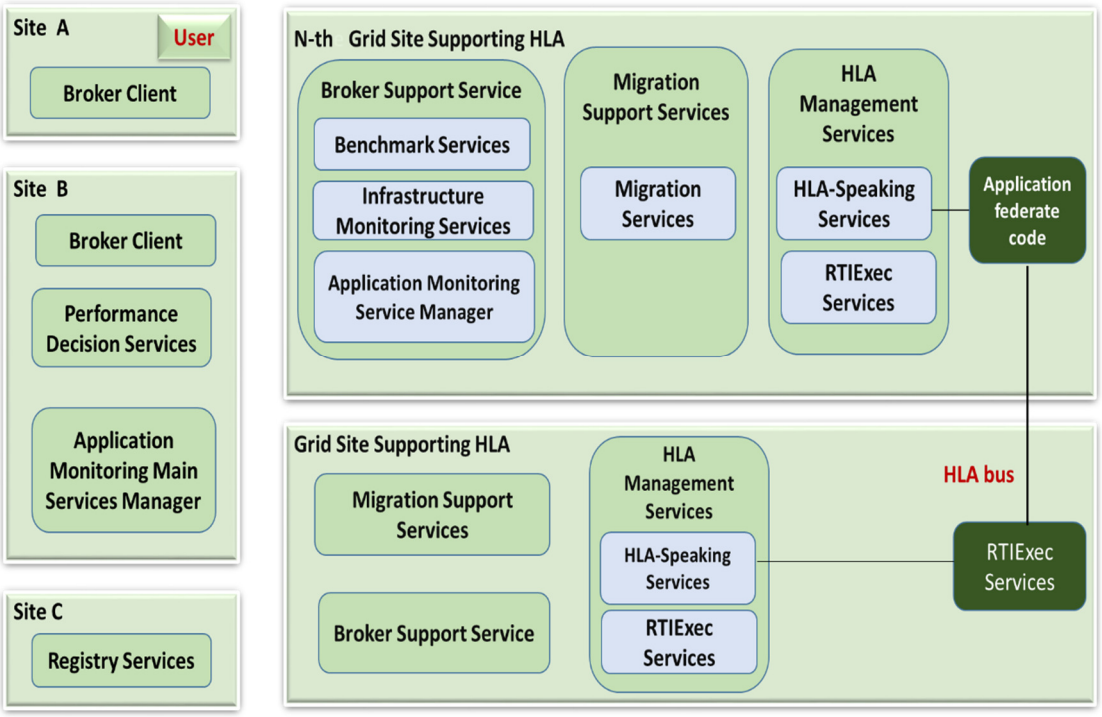


Figure 2.5 The G-HLAM framework *[20]*

The Broker Service is used to perform the simulation management and determine the best resource to receive a federate [28]. The Performance Service is used to manage the load redistribution. The Registry Service stores the information about local services that perform migration.

2.3.5 HLA-GRID

The author of [42] devised a new distribution simulation framework called HLAGrid. HLAGrid uses a Federate-Proxy-TRI architecture. This architecture was created to allow resources on a grid to be utilized on demand by using grid services. The Federate-Proxy-TRI architecture supports federation discovery, the security of the simulation logic, and flexible federation construction.

2.3.6 Grid-based Distributed Simulation Architecture

The authors of [27] have proposed a new Grid-based Distributed Simulation Architecture (GDSA). GDSA supports QoS-based scheduling and fault-tolerant computing. RTI services are employed in this architecture. GDSA and RTI have improved HLA simulation performance. As stated in [27], "GDSA, a full grid-based architecture mainly focuses on four pending problems in distributed system: scalability, communications, management mechanism and QoS insurance computing environment." The authors have added a Contractual Computing Mechanism (CCM) that provides QoS insurance for the user. Also, the QoS model divides the problems into three levels in the treatment of distributed simulation: a Base-Level QoS Guarantee, Service-Level QoS Guarantee, and System-Level QoS Guarantee [27].

2.3.7 Grid-Based Parallel and Distributed Simulation Environment

The authors of [43] [44] presented a Grid-based Parallel Simulation environment (GPDS). The GPDS addressed distributed simulation issues such as deficient computing powers, weakness in fault, and security issue. The GPDS used grid technologies to support transparency and scalability.

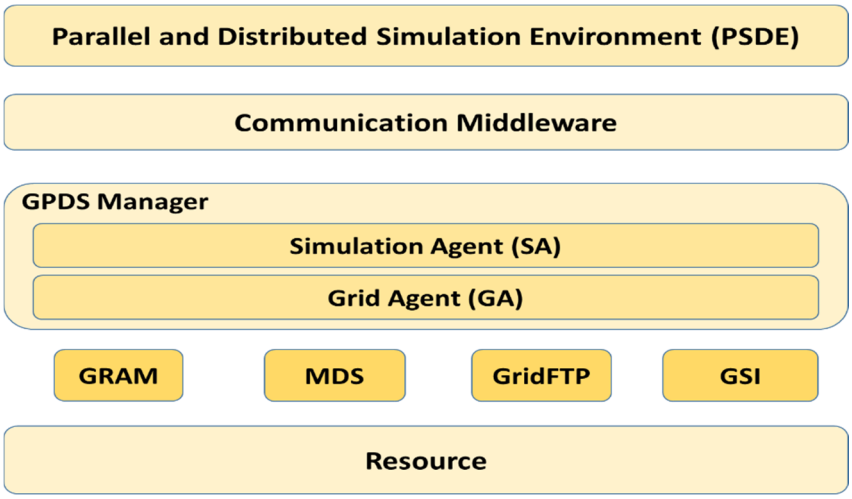


Figure 2.6 GPDS Architecture *[20]*

Three services have been proposed: automatic distribution service, dynamic migration service, and security service to avoid issues such as deficient performance or computing power as well as issues that are related to faults and security. This service has been divided into a three-tier architecture: clients at the front end, interaction server at the middle, and a network of computing resources, including databases, at the back-end. As shown in Figure 2.6 and Figure 2.7, the architecture consists of the simulation system, the communication middleware, GPDS manager, grid services, and resources. HLA and RTI are employed as simulation middleware to provide stable communication and interoperability. Therefore, the GPDS manager allows the Parallel and Distributed Simulation Environment (PSDE) and grid computing to simplify interactions with each other by using a Grid Agent (GA) and Simulation Agent (SA). However, it has not yet been presented how the federate state is saved or restored.

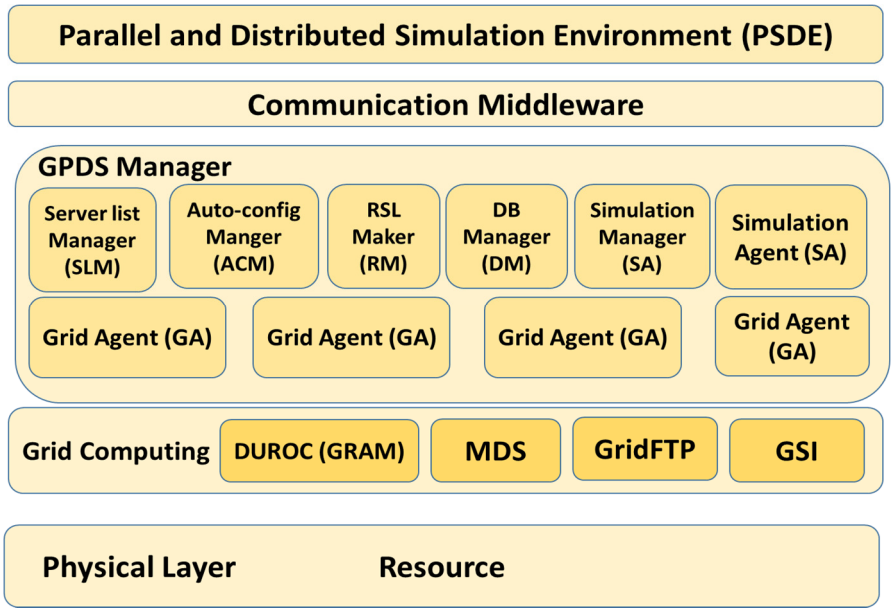


Figure 2.7 GPDS Layer *[20]*

2.3.8HLA GRID REPAST

Authors in [45] [46] proposed a HLA_GRID_REPAST which has been presented to manage large-scale distributed agent-based simulation by using a grid system. This works as a middleware between parallel and distributed elements of a simulation to perform the communication between them. The HLA_GRID_REPAST consists of HLA_REPAST [45] [47] and HLA_GRID [7] [48]. The HLA_REPAST is a middleware that supports agent-based distributed simulation with HLA based services. The HLA_GRID has been proposed as an architecture for employing HLA simulation over grid systems.

2.3.8.1 RePast System

The RePast system is a Java-based toolkit [49] that has been used for managing and developing the simulation based on agent model [45] [46] to provide several mechanisms and structures.

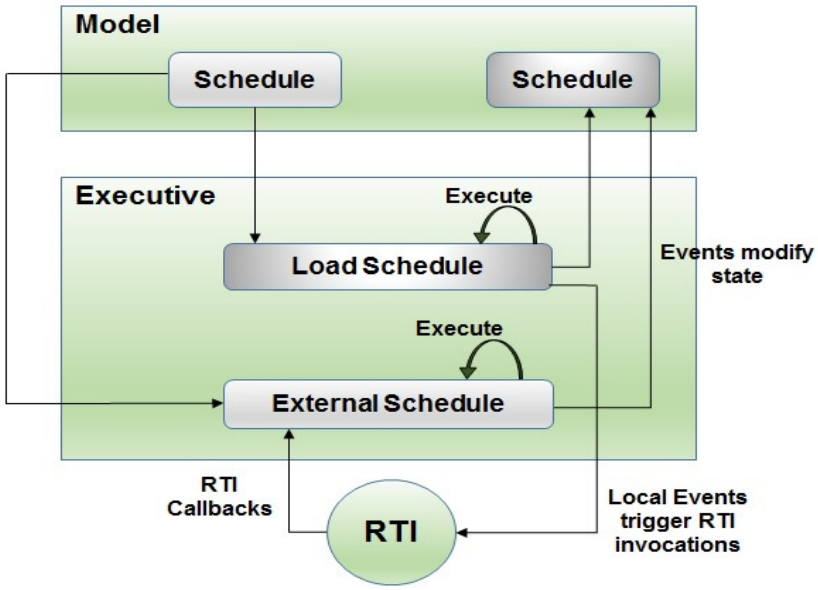


Figure 2.8 HLA_RePast Architecture *[20]*

2.3.8.2 HLA_RePast

The authors in [46] used the RePast agent-based toolkit that allowed the HLA_RePast to manage the simulation over distributed resources. The toolkit acts as middleware to introduce the connection between the sequential agent-based simulation system and the HLA framework. As shown in Figure 2.8 and Figure 2.9, there are several HLA_Repast components that interact using RTI. The execution of federations is managed using the RePast middleware.

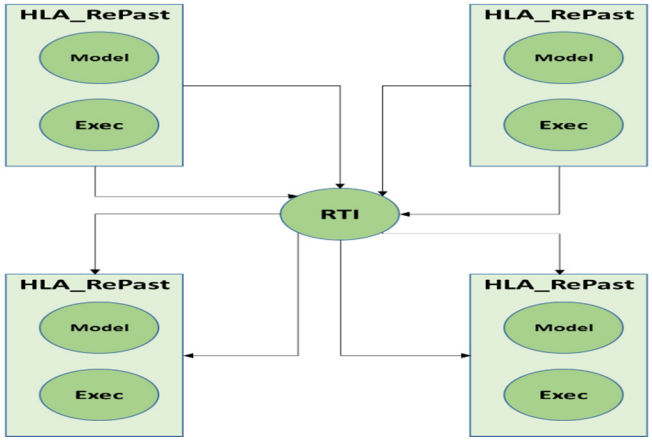


Figure 2.9 HLA Management Model *[20]*

2.3.8.3 HLA GRID REPAST

The authors of [47] [46] proposed the HLA_GRID_REPAST that can run and manage large-scale distributed agent-based simulations on Grid Systems. The HLA_GRID_REPAST is composed of integrating HLA REPAST and HLA GRID and acting as middleware. The HLA architecture is divided into a proxy side and a client side. As shown in Figure 2.10, the proxy side contains the Proxy RTI Ambassador Service and the Proxy Federate Ambassador. Therefore, the client side contains REPAST agent-based simulation system, HLA REPAST, Client RTI Ambassador, and a Client Federate Ambassador Service.

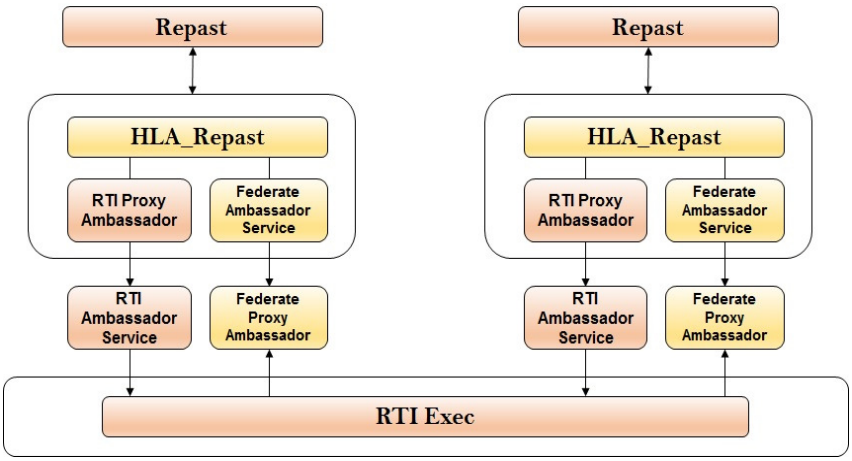


Figure 2.10 HLA_GRID_REPAST Architecture *[20]*

2.3.9 Service Oriented HLA RTI

The authors of [50] proposed a new framework called Service Oriented HLA RTI (SOHR). The SOHR framework has implemented the HLA/RTI using grid services. The authors classified the HLA-based simulation on the grid into three approaches: grid-facilitated, grid-enabled, or grid-oriented. However, SOHR was classified as grid-oriented. SOHR employed the six HLA RTI management services: Federate Management Service (FMS), Declaration Management Service (DMS), Object and Ownership Management Service (OOMS), Time Management Service (TMS), and Data Distribution Management Service (DDMS). This was proposed to improve the flexibility and interoperability between simulations.

2.3.10 Summary

All systems and architectures that have been mentioned above aim to support large-scale HLA-based simulation by using Grid Services. Most of the architectures are used to enable load-balancing and fault-tolerance mechanisms for such simulations. However, some schemes do not use any load-balancing mechanism, or are not defined, or the description was insufficient.

Chapter 3. Related Work

This chapter presents the load-balancing approaches that have been attempted previously. Examination of related work will demonstrate how these approaches have dealt with coordinating federate migration and load balancing in the distributed simulation. In addition, the descriptions will provide information about challenges that have been identified and describe the drawbacks of existing solutions.

The load-balancing approaches have two main streams: balancing systems for distributed simulations and federate migration protocols.

3.1. Balancing Systems for Distributed Simulations

The most important goal of the dynamic load-balancing system is to improve the simulation performance. This system has been proposed in many load-balancing approaches to observe the uneven load-partitioning issues for Parallel Discreet Event Simulation (PDES) distributed on shared resources, and to evenly divide the load partitioning [51] [52] [53]. The systems in these approaches have attempted to identify load imbalances, decrease simulation time, and perform load transfer. Most balancing approaches have been considered for many aspects in their design, including resource heterogeneity, external background load, monitoring metrics, simulation computing load, simulation entities interactions, as well as other simulation-specific characteristics, such as look ahead and virtual time progress. However, a solution has not yet been discovered that is suitable for HLA simulations running on a large-scale distributed system.

3.1.1 Balancing Schemes for Optimistic Simulations

According to [50], [53], and [54], schemes have been designed based more on the characteristics of parallel simulation rather than on optimistic simulation. In optimistic parallel simulation, schemes have been utilized to decrease simulation execution time by using specific characteristics. These schemes have identified the load imbalances and defined the rearrangement of simulation entities. The following characteristics have been used in the balancing system: Virtual Time Progress (VTP), Global Virtual Time (GVT), or Least Virtual Time (LVT).

The authors of [12] have proposed a new metric using simulation advance rate. This metric evaluates the imbalances of an optimistic distributed simulation. The rate is based on CPU allocation and simulation time advance of the previous request for each processor. The CPU allocation represents the time that has been consumed. The simulation advance "is the amount of time the local simulation clock has advanced" [55]. Based on this metric, the nodes that have greater advance time are elected to receive more loads based on node capacity. However, according to [55], this scheme does not support a heterogeneous system. Instead, the authors of [55] extended their metric to consider the heterogeneous system, and have considered a non-detectable system. However, neither scheme considers the external load.

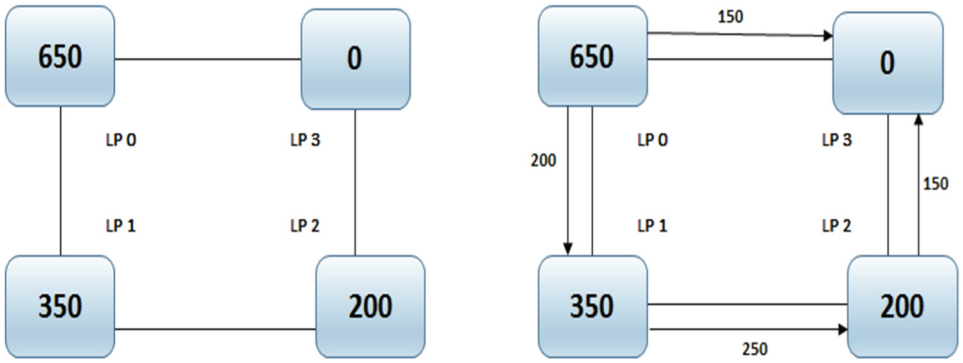
The authors of [56] developed a new dynamic load balancing scheme for optimistic simulation based on the Last Virtual Time (LVT) of each resource. This scheme uses static and dynamic load balancing. It was observed that the dynamic load balancing performed better when the loads were heavy during the distribution. However, this scheme does not support heterogeneous and non-detectable systems.

The authors of [57] proposed a load-balancing scheme based on two techniques: dynamic load balancing technique and static partitioning technique. The dynamic load-balancing technique uses the Virtual Time Progress (VTP). The VTP is responsible for representing the average simulation speed in a processor. The static technique is responsible for grouping the partitioning in a cluster based on the detection of strongly connected regions. However, this scheme does not support heterogeneous and non-detectable systems.

The authors of [58] devised a new scheme called dynamic load-balancing in parallel discrete event simulation (PDES) for spatially explicit problems. This scheme is modeled as a ring. As shown in Example 1 and Example 2, each node considers Logic Processes (LPs). Using GVT calculation that detects the load imbalances in every collection will identify the number of unprocessed events. The dominant load is responsible for identifying the imbalances. Thus, the dominant ring will re-distribute the load evenly between neighbors. Through the process of re-distribution, certain loads will migrate from the overloaded LP to the underloaded LP, as shown in examples 1 and 2. The overloaded LP will back-up the information before migration occurs. However, this scheme does not support the heterogeneous resources.

The authors of [59] presented a flow control and dynamic load-balancing scheme. This scheme was employed to improve performance and stability. The flow control algorithm was used to organize the flow of messages that occurs between processors. The dynamic load balancing is not employed until load imbalances have been detected. The migrations occur when there is an imbalance from the overloaded to the underloaded

federates. However, this scheme does not consider non-detectable resources or heterogeneous resources.

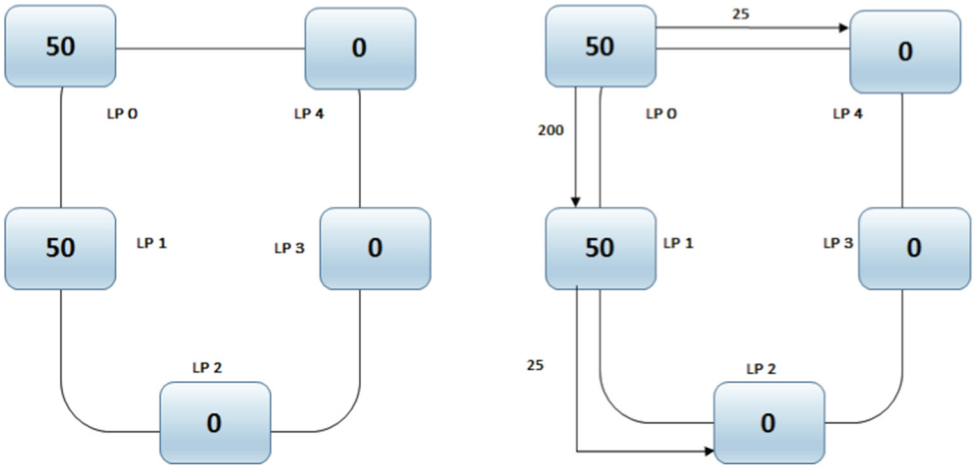


(a) Original Load Graph (b) Load Balancing

Figure 3.1 Example 1 *[20]*

According to [60], a new scheme was designed called dynamic load management with a cost model. This model was developed based on communication rate, computation load, and lookahead. However, the author did not explain the load transfer technique. Also, this scheme does not consider heterogeneous or external loads.

The authors of [61] proposed a new mechanism, called dynamic load balancing for optimistic simulation that considers communication and computation imbalances. The computational and communication load metric is employed to monitor communication and computation imbalances. The scheme is similar to [12], in that time advance is used to identify the overloaded processes. Following detection, the overloaded nodes will redistribute the load by transferring a certain load from the overloaded nodes to the underloaded nodes.



(a) Original Load Graph (b) Load Balancing

Figure 3.2 Example 2 *[20]*

3.1.2 Balancing Schemes for Conservative Simulations

Another load-balancing scheme has been developed for Conservative Simulations [53] [62] [63]. The metrics used in this balancing system have been previously mentioned in the balancing schemes for Optimistic Simulations such as lookahead to detect load imbalances and certain metrics that can monitor the current processors and communication load for shared resources. These metrics can redistribute the load dynamically.

The authors in [64] devised another balancing system for conservative simulations. This scheme attempts to improve the simulation performance by decreasing the number of null messages and distributing the load evenly. The load of LPs metric is based on CPU-queue length utilization. This is compared to the threshold to distinguish between the overloaded and underloaded nodes. However, this scheme did not consider the heterogeneity aspects nor the external loads when the simulations are redistributing the loads.

The authors of [65] introduced a new Critical Channel Traversing (CCT) algorithm. This scheme proposes a new scheduling algorithm for Conservative Parallel Discrete Event Simulation. In this scheme, a centralized queue stores the tasks in the queue consecutively. Rescheduling occurs based on CCT selecting a task from the queue to run in LP or a set of LPs. Furthermore; the balancing load distribution minimizes the communication and computational delay. However, this scheme did not resolve issues that occurred due to heterogeneity or non-deducted resources.

The authors of [66] proposed a new scheme for dynamic load balancing system for conservative distributed simulations, which runs on a shared memory multiprocessor system. This system employs static partitioning and dynamic load balancing: "The static partitioning scheme maps simulation objects to logical processes before simulation starts while the dynamic load balancing scheme attempts to balance the load during runtime" [66]. Furthermore, the static partitioning scheme attempts to decrease the load imbalances while increasing the lookahead values. However, this scheme does not support heterogeneous or external loads, nor does it resolve scalability issues.

The authors of [67], [68], and [69] devised a new scheme of a dynamic load partitioning system for conservative distributed simulations called Biocomputing techniques that run on distributed environments. In these schemes, a heuristic technique is employed to find suboptimal solutions. Furthermore, the authors of [70] proposed a simulated annealing technique with an adaptation that can help to find the sub-optimal solution. These schemes consider the load and the communication load for each processor calculation. However, the authors do not mention how to monitor when it is that migration has occurred. Moreover, these schemes do not support heterogeneity or non-deducted resources.

3.1.3 Balancing Schemes for HLA-based Simulations

Balancing schemes have been developed to perform HLA-based simulations. The present architecture for performing the schemes has been employed in the HLA framework. This combination enables the transparency transferring of federates during migration.

The authors in [30] proposed a new design called Resource Sharing System (RSS). The RSS is used to provide a dynamic load-balancing system for HLA-based distributed simulation. The RSS scheme controls the availability in their system for a distributed simulation mechanism. Although federate migration mechanism has been described, there is not enough information about other load-balancing aspects. The system architecture consists of three parts: an RSS manager, communication federate, and accessing of a FTP Server. Furthermore, during the migration transfer, the simulation is globally synchronized. However, this balancing scheme is not suitable for fault-tolerance; moreover, it introduces the high-latency federate migration mechanism.

The authors of [34] developed a Load Management System (LMS) that supports HLA-based distributed simulation of geographically different organizations with efficient and effective simulations. The LMS used a Globus toolkit [35] [36] that has been devised to enable Grid Computing. The toolkit manages all resources on distributed simulation through this system. Therefore, the RTI supports and manages the execution of the simulation federation. During the Reliable Data Transfer (GridFTP) that is responsible for transmitting data, the system stops the simulation. After that, the execution of jobs begins remotely by accessing the Grid Resource Allocation Manager (GRAM). The most important element of this scheme is that it considers the external loads through the GIS. However, this scheme does not support heterogeneity aspects.

The authors in [33] designed a new resource management system based on Grid services. This system supports the execution of interactive HLA-based simulations. The system employs Grid services to allow configuration of a dynamic simulation. During the run-time, the Grid services are used to transfer the data for federate migration. During the federate migration transfer, the simulation is globally synchronized, and the system stops the simulation. By using the GridFTP, data is transferred reliably. However, although this scheme introduces the high-latency federate migration mechanism, it does not introduce a monitoring and re-distributed mechanism.

The authors of [71] devised a new load-distribution technique for HLA-based simulation. This scheme focuses on optimistic federate migration, and introduces an interface component called a Federate Wrapper. The Federate Wrapper is employed to re-distribute the simulation load, perform federate migration, and conduct monitoring. In the federate migration, a mechanism has been proposed with three queues to avoid simulation inconsistencies and minimize delays. Furthermore, this scheme is based on time advance. However the scheme does not support heterogeneity aspects or load re-distribution.

3.1.4 Summary

Table 3-1 presents a comparison of what has been proposed with some limitations. These are the balancing systems of discrete-event simulations that can be utilized in large-scale distributed systems.

Table 3-1 Comparison of Balancing Schemes for Discrete-Event Simulations *[20]*

Proposed by	Sim.	Monitoring	Re-distribution	Migration	Heterogeneity	External Load
Glazer & Tropper (93)	Opt	Time Advance	Comp.	-	-	-
Burdorf & Marti (93)	Opt	LVT (Vector)	Comp. Speed (StD)	Simplistic (Slow)	Partially (Indirectly)	Partially (Indirectly)
Jiang et al.(94)	Opt	Time Advance	Comp.	-	Weights	-
Schlagenhaft et al. (95)	Opt	VTP	Comp. pVTP + Mig.	Undefined	-	-
Avril & Tropper (96)	Opt	Comm. Throughput	Load (Comm.)	Undefined	-	-
Carothers & Fujimoto (96)	Opt	PAT, TWFrac	Load (Policies)	Clustered	Yes	Partially (Limited)
Wilson & Shen (98)	Opt	CPU Load	Policies (Comm/Comp)	-	-	-
Deelman & Szymanski (98)	Opt	Unproc. Events	Comp. (Chains)	Neighbour	-	-
Choe & Tropper (99)	Opt	Space-Time Prod.	Comp.	Undefined	-	-
Low (02)	Opt	CPU load	Comm. Comp. Lookahead	-	-	-
Jiang et al.(04)	Opt	IPC	Comp. + Comm.	Clustered (Slow)	-	-
Peschlow et al. (07)	Opt	Time Advance	Comm. Comp.	-	-	-
Xiao et al. (99)	Con	Comm. Dep.	Scheduling lvl	-	-	-
Gan et al. (00)	Con	Sim. Time	Central (Priority)	-	-	-
Boukerche (04)	Con	Entropy	Comp. + Comm.	-	-	-
Ajaltouni & Zhang & Boukerche (08)	Con	CPU Load	Comp. + Comm.	Global Sync	-	-
Luthi & Grossmman (01)	HLA	-	-	Global Sync.	-	-

Cai et al. (02)	HLA	Grid	-	Global Sync.	-	Only Monitoring
Zajac et al. (03)	HLA	Grid	-	Global Sync.	-	Only Monitoring
Tan & Lim (05)	HLA		-	Queues		
Robson & Mohammed & Boukerche (12)	HLA	Grid	Comp. + Comm.	Grid & Peer to Peer	Yes	Aware (Grid)

3.2. Federate Migration Mechanisms

Migration is common in several areas which use migration mechanisms such as process migration and mobile agent migration in parallel and distributed computing. The authors in [72] proposed a new technique that is Java-based on a mobile agent. This technique has been divided into two parts: execution code and data. The mobile agent scheme is proposed for its ability to migrate autonomously, adapt to different environments, and seamlessly recover the execution state. The most important aspect of mobile agent schemes is interoperability, which has improved the transparency in distributed systems by minimizing the migration effects on other systems.

The authors of [73] proposed a process migration. The process migration is comprised of three phases: negotiation, transfer, and establishment. The negotiation phase controls the decision of whether to transfer or not transfer, based on the agreement that has been received from the process. The transfer is responsible for copying the communication link and processing the address from the source to the destination. This initiates the transfer to the destination.

Federate migration approaches are a blend of process migration and mobile

agent techniques. Using both techniques creates the advantage of process migration and incorporates the interoperability and transparency features of mobile agents.

The authors in [30] developed a new solution for large-scale distributed simulations by using HLA with RSS. The solution for simulation systems is divided into two parts: a simulation manager and communication federate. Both are employed on migration processes. However, the entire simulation freezes when the communication federate transmission begins.

The authors in [33] designed a new resource management system based on Grid services. This system, which supports the execution of interactive HLA-based simulations, proposed a Migration Library between the HLA simulation and Grid services. The Migration Library is an interface to simplify the HLA Application Programming Interface (API). By using the GridFTP, data is transferred reliably. The federate state is saved and restored by using the HLA specification method.

Authors in [34] devised a new Load Management System (LMS) to support HLA based on large-scale distributed simulation. The RTI supports and manages the execution of the simulation federation. Furthermore, this system supports federate migration. However, the entire simulation freezes when the federate migrations that use the GridFTP are started.

The authors of [71] devised a new load distribution technique for HLA-based simulation. This scheme focuses on optimistic federate migration and introduces an interface component called a Federate Wrapper and a load distribution system. The Federate Wrapper controls the federate execution. The load distribution system is responsible for monitoring federates and making decisions related to migration time. The federate migration

mechanism proposes three queues to avoid simulation inconsistencies and minimize delays. The queues store the messages during migration.

The authors in [74] proposed a new migration technique based on SimKernel. The SimKernel is employed to minimize migration latency by considering the application-level federate migration. The most important aspect of this scheme is that no third-party mechanism is used, and there is no freezing when the federate migrations begin.

The authors of [75] developed an HLA federate migration. The HLA federate migration design is comprised of Federate, fedMonitor, and fedServer. The Federate is a simulation component, the fedMonitor is responsible for controlling the federate migration, and the fedServer supports the federate migration. The data transfer uses peer-to-peer communication. The most important advantage of this mechanism is that it does not freeze when the federate migrations begin.

3.3. Migration-Aware-based Dynamic Load Balancing System

Most load redistribution schemes consider the connectivity between load imbalances (computation and communication) and available resources. This consideration allows the scheme to make a decision to redistribute a load when necessary. However, the migration latency directly affects the load distribution performance. Some schemes have considered migration latency, but not in depth, such as by evaluating the migration delay in their simulation performance. However, the authors of [15] proposed a new scheme that considers migration delay by measuring and analyzing the migration delay in the simulation.

The measurement and analysis of migration delay develops the distributed balancing system on the load re-distribution algorithm. This scheme includes an estimation based on the past simulation load migration because the real migration delay values are not obtained until the migrations are completed. When the real migration delay values are obtained, these values enable to estimate and analysis for the next migration cost. Furthermore, the decision making for the next migration is calculated based on the real migration delay values. Subsequently, the schemes are employed on the migration metric to measure and monitor the recourses, which results to evaluate the resources before the migration call is issued.

3.3.1 General Architecture

The proposed scheme introduces migration awareness. As shown in Figure 3.3, the main component in the proposed balancing system is the Cluster Load Balancer (CLB). The CLB is responsible for coordinating all other balancing components and leading the balancing procedure. The CLB has two parts—simulation and recourses—to collect the measured load and state the data. The CLB obtains the information for the load status from the Monitoring Information Service (MIS). The MIS is responsible for retrieving the load status data by using a third-party tool. The Grid Services are employed to present the processor queue length of each resource. Furthermore, the Ganglia and HawKEye are third-party monitoring tools to provide information for the balancing system by retrieving the state of the resource load.

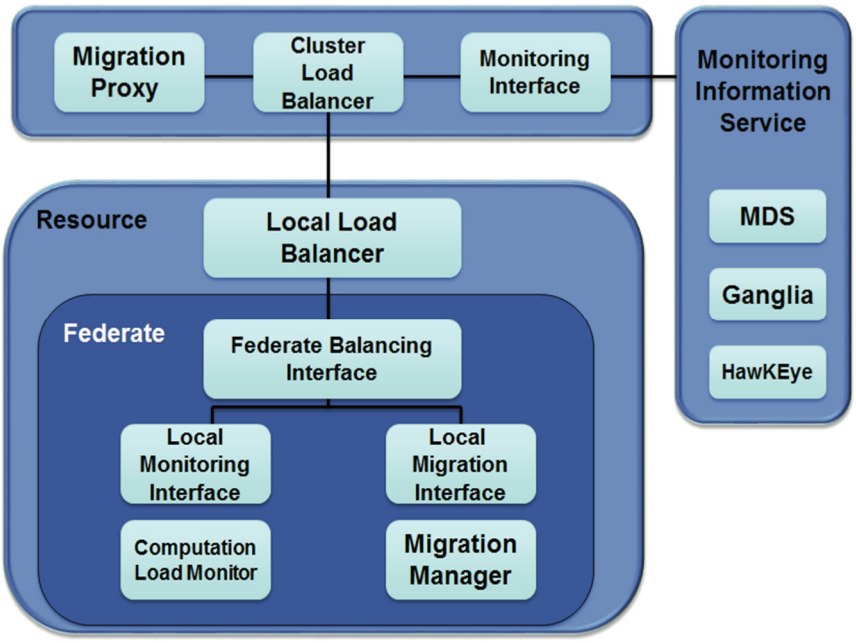


Figure 3.3 Migration-Aware Load Balancer System Architecture *[20]*

The Local Load Balancer acts as an interface between the balancing system and virtual simulation. LLB collects, aggregates, and sends the information for each federate load consumption to the CLB. The Computation Load Monitor tracks the amount of time that has been used by the CPU for each federate. The Local Monitoring Interface is employed to gather the load information of each federate during the balancing time.

From an analysis perspective, the CLB collects the load information of each federate and measures migration. The Migration Manager (MM) is used to coordinate the entire federate migration procedure. The MM is responsible for the transfer of federates between resources without losing any data. Furthermore, MM uses a two-phase migration technique [76] [77]: transfers of static information and transfers of dynamic data.

Transfers of static information are used to transfer the data reliably by using Grid services (third-party tool). The dynamic data is used to transfer the dynamic execution information between MMs by using a peer-to-peer technique.

3.3.2 Balancing Algorithm

The balancing algorithm is used to rearrange the federate load in the distributed simulation. As shown in Algorithm 1, the balancing technique is initiated in cycles to collect the data. The migration-aware balancing scheme is divided into three phases: monitoring, re-distribution, and migration. The monitoring phase is responsible for detecting the load imbalances that occur in the current distribution of load state.

The metrics of computational load are employed to observe the redistribution analysis. The re-distribution phase uses a greedy mechanism to enable a global sub-optimal balancing solution. The migration phase is used to coordinate the entire federate migration procedure.

Algorithm 1 : Main Load Balancing Algorithm [20]

```
while TRUE do
        loads ⇐ query MDS()
        current loads ⇐ filter MDS data(loads)
        current loads ⇐ normalize loads(current loads, benchmark)
        overload cand ⇐ select overload(current loads)
        spec loads ⇐ request LLBs(overload cand)
        mng loads ⇐ filter(current loads, spec loads)
        mean, bds ⇐ calculate mean bds(mng loads)
        over, under ⇐ select(mng loads,mean, bds)
        mig moves ⇐ redistribute local(mng loads)
        mig moves ⇐ analyze migration latency(mig moves)
        send migration moves(mig moves)
        if mig moves = ∅ then
                data neighbours ⇐ request Neighbour Load Data()
        else
                if    relFactor ≥ random number(1, 100) then
                      data neighbours ⇐ request Neighbour Load Data()
                else
                       data neighbours ⇐ ∅
                end
        end
        wait( Δt )
end
```

The distributed dynamic load-balancing system periodically monitors the resources. This allows the balancing system to be aware of when the load is changed. By using the monitoring tool, the monitoring phase is triggered periodically every Δt, which is limited by the cyclic refresh rate. The balancing system settles in 20 seconds.

Once the CLB has aggregated the information that was sent from the LLB and MIS, as shown in Algorithm 2, the CLB divides the status of the federates into two lists: overloaded federates and underloaded federates. Afterwards, the most overloaded federate is matched with the least underloaded federate.

Algorithm 2: Pair-Match Evaluation Algorithm [20]

```
Require: src rsc, dst rsc
selected federate ⇐ select federate smallestLatency(src rsc)
if dst rsc < min then
        if number fed(src rsc) ≥ 1 & src rsc > (min ∗ ϕ) then
                Δt′ ⇐ Δt × α
                create migration move(src rsc, dst rsc, selected federate)
        else if number fed(src rsc) > 1 then
                Δt′ ⇐ Δt
                create migration move(src rsc, dst rsc, selected federate)
        end
else if (dst rsc − src rsc) > (min ∗ δ) then
        if number fed(src rsc) ≥ 1 AND (dst rsc − src rsc) > (min ∗ ϕ) then
        Δt′ ⇐ Δt × α
                create migration move(src rsc, dst rsc, selected federate)
        else if number fed(src rsc) > 1 then
                Δt′ ⇐ Δt
                create migration move(src rsc, dst rsc, selected federate)
        end
end
if migrationMove then
        estimatedGain ⇐ estimateMigGain(dst rsc, src rsc,Δt′)
        estMigTime ⇐ estMigTime(dst rsc, src rsc, selected federate)
        Return: migrationMove, estimatedGain, estimatedMigTime
End
```

$$\mathbf{t_e} = \frac{\mathbf{time_{mig}}}{\mathbf{dist_{mig}}} * \mathbf{dist_{dst}} \quad \textbf{(3.1)}$$

t_e: estimation of migration delay
$time_{mig}$: time of past migration
$dist_{mig}$: distance of past migration
$dist_{dst}$: distance of estimated migration

Based on this matching collection, the migration delay estimation (t_e) and the estimation of time gain (t_d) are both applied for each match, as shown in formulas (3.1) and (3.2) [13]. Based on these formulas, a comparison is made between t_d and t_e , and the match will be deleted if t_e is greater than t_d. As shown in Algorithm 3, once the comparison has any calculation showing that t_d is greater than t_e, then the migration move list is created.

$$\mathbf{t_d} = \frac{\Delta \mathbf{t} \times (\mathbf{load_{src}} - \mathbf{min_{load}} - \mathbf{load_{dst}})}{\mathbf{load_{src}}} \quad \textbf{(3.2)}$$

t_d: estimation of time gain
$load_{src}$: current load of source resource
min_{load}: estimated minimum load
$load_{dst}$: current load of destination resource

Algorithm 3: Migration Latency Filtering Algorithm [20]

```
Require: mig moves
If mig moves! = ∅ then
        timeGain ⇐ calculateGain(mig moves)
        expectedMigrationDelay ← calculate(mig moves)
        while timeGain ≤ expectedMigrationDelay do
                mig moves ⇐ eliminate smallest gain(mig moves)
                timeGain ⇐ calculateGain(mig moves)
                expectedMigrationDelay ⇐ calculate(mig moves)

        end
end
Return: mig moves
```

Furthermore, the Δt is employed as a time factor. The Δt is used to observe the simulation performance. Once the matching filter is complete, then the migration move list is created. Afterwards, comparisons are applied between the accumulated final estimations of performance gain and migration latency, as shown in formulas (3.3) and (3.4) [15].

$$t_{d_s} = \sum_{i}^{n} t_{d_i} \tag{3.3}$$

t_{ds}: overall sum of migration time gains;
t_{di} : time gain of each migration move.

Filtering is repeatedly applied when the time gain is smaller than the estimation delay. The final migration move set is created when t_{ds} is shown to be greater than t_{es}. Afterwards, CLB sends a migration call to MM to start the migration moves of the federates.

$$t_{e_s} = t_{e_{largest}} + \frac{\alpha * \sum_{i \neq e_{largest}}^{n} t_{e_i}}{(n-1)} \tag{3.4}$$

t_{es}: overall estimation of migration delays
$t_{elargest}$: largest migration delays in the set of migration candidates
α: influence of other migration delays on overall estimation
t_{ei} : delay of a migration move candidate
$e_{largest}$: migration move with the largest delay

Chapter 4. Enhancing Load-Balancing Estimations based on Migration Awareness

Migration latency affects load re-distribution performance. Latency has the responsibility of determining which HLA federates can be moved for better performance. In terms of load imbalances, the relation between computational, commutation, and shared resources is the key to redistributed load. The decision to migrate federates is based on this information.

This balancing system is employed in the load-redistributed algorithm to allow the system to master and analyze the migration latency. Based on this information, the calculation for the estimation of migration delay is measured and analyzed to determine the cost of migration delay. Thus, this estimation is very important for good simulation performance. Information that has been collected from past migration, which is a real migration value, is provided to estimate the next migration.

4.1. General Architecture of the Enhancing Load-Balancing Estimation System

The enhanced load balancing architecture is based on the migration-aware dynamic balancing system [15], and the proposed balancing scheme system algorithm is similarly defined as in Figure 3.3, adding the *nload* tool. The *nload* is a tool that monitors the network traffic and bandwidth usage in real time. Also, it provides additional information

such as incoming and outgoing traffic: minimum, maximum and average network usage. Furthermore, this tool visualizes the total amount of data that has been transferred through the network resources. Consequently, we have used two real time values that have been recorded by nload tool. These values are the current bandwidth and average bandwidth (as historical usage) to estimate the time gain in our estimation. The Cluster Load Balancer (CLB) is the main part of the balancing system, and is responsible for coordinating between the other balancing parts. Also, the CLB leads all the tasks for the balancing scheme procedure.

The CLB needs to know the load status data for both the simulation part and the resource part. Thus, the CLB requests the resource load status by accessing the Monitor Interface. As depicted in Figure 4.1, the Monitor Interface receives the request from the CLB. Afterwards, the Monitor Interface requests the resource load status by sending a request to the Monitor Interface Services (MIS). The MIS retrieves the load status for each resource by using third party monitoring tools. The MIS employs the Grid Services [6] that can provide the processor's queue length for each resource. Therefore, the CLB receives and saves the data that has been aggregated on the Monitoring Interface from the MIS.

The Local Monitoring Interface (LMI) is employed to provide the federate load of virtual simulations. The Local Load Balancer (LLB) is an interface between the Virtual simulation and the balancing system. The LLB accesses each federate which allows management of the federates; in addition, the LLB aggregates the data that have been sent by the LMI. When the CLB needs to check the load balancing, a load metric is used to detect any load imbalances by the information that has been collected from Monitoring Interface

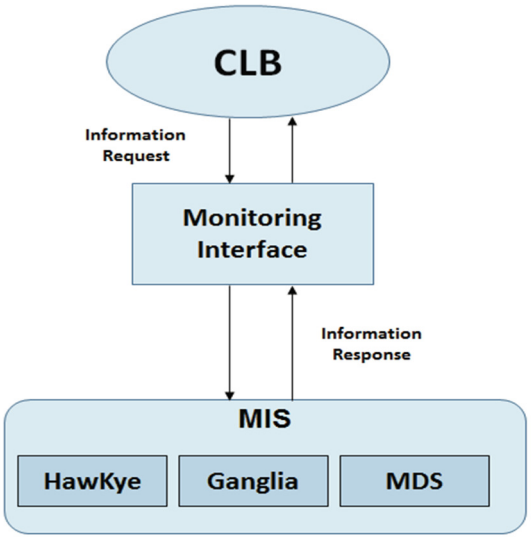


Figure 4.1 CLB resources load status request

and the LLB. The Computation Load Monitor (CLM) tracks the amount of time that has been used by the CPU for each federate. The Local Monitoring Interface is employed to gather the load information of each federate during the balancing time.

As shown in *Figure 3.3*, the MIS has an additional nload tool. The nload tool provides the amount of the link capacity, specifically how much bandwidth the destination resources currently use. By accessing MM, the Local Migration Interface (LMI) provides the receiving and finishing time of the destination resource that was received from the source resource migration data. The CLB collects the bandwidth usage information and the arrival and finish time information for each federate, and measures the migration move. Based on this information, the migration call is initiated and sent to the Migration Manager (MM). Once the MM receives the migration call, the MM coordinates the entire federate migration procedure. Thus, the MM is responsible for the transfer of federates between resources without loss of any data. Consequently, the MM uses a two-phase migration technique: transfers of static information and transfers of dynamic data. Transfers of static information are used to transfer the data reliably by using Grid services (third-

party tool). The dynamic data is used to transfer the dynamic execution information between MMs by using a peer-to-peer technique through the migration proxy.

4.2. Proposed Solution

The proposed enhancement for the estimation in [15] is divided into three scopes: Δt, estimation time delay, and estimation time gain. The Δt is the cycle time that is used as a parameter that defines and compares the performance for the simulation over a short time period. The estimation time delay is the amount of time taken to migrate the federate from the source resources to the destination resources. The estimation of time gain is the amount of time taken to process the destination resources.

4.2.1 Flexible Δt

As described in Chapter 3, Δt is a time factor. There are two types of Δt. First type that is related to the simulation system, and it is unpredictable because there is no mechanism that can provide a precise value that tells the period a simulation application will run. The second type is related to the balancing system, and it is restricted by the balancing cycle interval, which comprehends 20 seconds for this particular migration-aware balancing system. Thus, it is challenging for it to be flexible because there is no mechanism that allows Δt to have flexibility. Previous estimation migration delay [15] has been calculated based on the past migration time. In order for the proposed estimation technique to be flexible and more adaptable, it is not based on the past migration time. The flexibility of the Δt is limited to not being more than 20 seconds and not being less than 10 seconds. The proposed solution is molded according to the following equation:

$$\Delta t_i = \frac{\Delta t_{i-1}}{t_{mig}} * \frac{\sum_{j=0}^{i-1} \Delta t_j}{i} \quad (4.1)$$

Δt_i : Current flexible Δt
Δt_j : Previous Δt
t_{mig} : Previous migration delay
Δt = 20 sec
Range:

$$10 \leq \Delta t \leq 20$$

4.2.2Estimation of Time Delay

The proposed estimation of time delay is divided into two parts. As described in formula (4.2*)*, the first part is comprised of $Finish_{time}$ and $Arrival_{time}$ and is referred to as response time [78] to identify how long it has taken the message to be processed. This information is provided by the RTI/HLA component to determine when the Destination resources have received the messages, and when the destination resources have executed the messages that were received from the source resources by migrating data. The $Finish_{time}$ and $Arrival_{time}$ reflect the processing time in the destination resource. The second part relates to the bandwidth usage, meaning how much bandwidth is currently used. The bandwidth usage reflects the transmission availability.

$$t_e = \left(Finish_{time} - Arrival_{time}\right) * \left\{\left(\frac{\text{Current usage}}{\text{Link capacity}}\right)\right\} \quad (4.2)$$

t_e **:** estimation of migration delay
Finish $_{time}$:when the destination resource executed the migration
Arrival $_{time}$: when the destination resource finished receiving the migration
Current usage: how much bandwidth is currently used
Link capacity: maximum of link bandwidth

4.2.3 Estimation of Time Gain

The proposed estimation of time gain is divided into two parts. As described in formula (4.3*)*, the first part is based on Δt. Δt is used as fixed value (20 seconds) in the first estimation. Next, the correct load is sent to the source resource and destination resource without a minimum load. This prevents any migration from occurring without time gain. The td is used for the decision-making on time gain and time delay, so it employs t_d, which molded by the difference of load between the destination and source resources participating in the migration.

$$\mathbf{t_d} = \mathbf{\Delta t} * \left[\frac{\mathbf{load_{dis}}}{(\mathbf{load_{src}} - \mathbf{min_{load}})}\right] \quad \textbf{(4.3)}$$

$\mathbf{t_d}$: estimation of time gain
$\mathbf{load_{src}}$: current load of source resource
$\mathbf{min_{load}}$: estimated minimum load
$\mathbf{load_{dst}}$: current load of destination resource
$\mathbf{\Delta t}$ = Flexible Δt

4.3. Balancing Algorithm of the Enhancing Load-Balancing Estimation System

As stated earlier, the proposed scheme of Enhancing Load-Balancing Estimation is divided into three passes: monitoring, redistribution, and migration. These phases are divided to handle the issues of each phase separately. When the balancing system begins to collect the data to investigate whether there are any load imbalances, two schemes [79] [10] are employed to improve the load redistribution. By using these schemes, the dis-

tributed balancing system retrieves the data that is used to measure and conduct an analysis of migration latency. Furthermore, a greedy technique is employed that provides a quick response to the load imbalances. These two schemes and the greedy technique are employed in the re-distribution phase.

Algorithm 4 : Main Load Balancing Algorithm [15]

```
while TRUE do
        loads ⇐ query MDS()
        current loads ⇐ filter MDS data(loads)
        current loads ⇐ normalize loads(current_loads, benchmark)
        overload_cand ⇐ select overload(current loads)
        spec loads ⇐ request LLBs(overload_cand)
        mng_loads ⇐ filter(current_loads, spec_loads)
        mean, bds ⇐ calculate mean bds(mng_loads)
        over, under ⇐ select(mng_loads, mean, bds)
        mig_moves ⇐ redistribute local(mng_loads)
        mi_ moves ⇐ analyze_migration_latency(mig_moves)
        send migration moves(mig_moves)
        if mig_moves = ∅ then
                data neighbors ⇐ request Neighbor_Load_Data()
        else
                if    relFactor ≥ random number(1, 100) then
                      data neighbors ⇐ request Neighbor Load Data()
                else
                      data neighbors ⇐ ∅
                end
        end
        wait (Δt)
end
```

As described in Algorithm 4, the CLB sends {query_MDS ( )} a request to the monitor interface about load status for resources and simulation. The monitoring interface sends this request from the CLB to the MIS. Afterwards, The MIS accumulates the load status from each resource, and disregards the resources that do not participate in the sim-

ulation. The MIS aggregates and sends the information to the CLB through the monitoring interface. The CLB normalizes the load rate by using a benchmark to solve inherent issues caused by resource heterogeneity.

Based on the information that is received from the monitoring interface, the overload is selected (spec_*loads* ⇐ *request LLBs (overload_cand)*). The CLB then sends a request to the LLB to send back more details about the CPU consumption for each federate (spec_*loads* ⇐ *request LLBs (overload_cand)*). The overloaded ranking is based on the CPU consumption. The balancing system filters any overloaded resources that are not identified or that do not have any simulation entities.

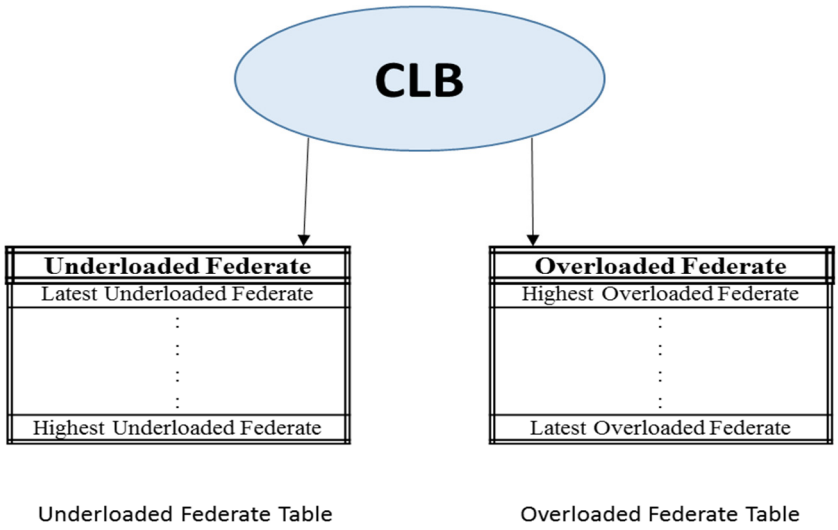


Figure 4.2 Create underloaded and overloaded lists

The balancing algorithm is divided into local and inter-domain parts. The local part's redistribution algorithm is triggered to generate the migration. The inter-domain is triggered based on the migration that is generated. The balancing system rearranges the local load to decrease load discrepancies between the resources in a domain. The balancing system calculates the mean and bds (*mean, bds* ⇐ *calculate_mean_bds*

(mng_loads)). Then, the balancing system analyzes (*over, under* ⇐ *select (mng_loads,* mean, bds)) their loads and creates two lists of overloaded and underloaded. As shown in Figure 4.2, the lists are ordered based on the state of each load. CLB matches the most overloaded with the most underloaded to obtain the highest benefit from this migration of overloaded to underloaded.

Algorithm 5 : Inter-Domain Redistribution Algorithm [15]

Require*: neighbors_data*
neighbors *identify_Neighbor_Less_Load()*
order_neighbors_by_load(selectionParameter)
if *neighbors* ≠ ∅ **then**
    neighbors ⇐*select_Neighbors(extStD, localStD)*
    order_neighbors _b_ load(selectionParameter)
end
for each *neighbor IN neighbors* **do**
    overloaded RSCs ⇐*select(neighbor)*
    federates ⇐*select(spec_loads, overloaded)*
end
if *overloaded RCSs* ≠ ∅ **then**
    sort list load(overloaded RSCs)
    sort list load(neighbor RSCs)
    moves ⇐*redistribute(overloaded RSCs, neighbor RSCs)*
    moves ⇐*analyze migration latency(moves)*
end
send migration moves(moves)
adjust factor(relFactor, overloaded RSCs, moves)

The CLB estimates the migration federates based on the information that has been gathered. However, if there is no migration federate (**if** *mig_moves* = ∅ **then**), the balancing system triggers the inter-doming load. As detailed in algorithm 5, the CLB requests the neighbor's information to calculate the local load and the neighbor's load, and then detects the large load differences. The balancing system identifies (neighbors ⇐ *identify_Neighbor_Less_Load ()*) the underloaded domain. Then, the balancing system

orders the overloaded neighbor from the highest to least. The underloaded neighbor CLBs and overloaded resources are selected according to Algorithm 5. The balancing system matches the underloaded neighbor CLBs with the overloaded resources. When all of the overloaded resources are assigned, then a migration move occurs. The balancing system analyzes the federate migration latency (*moves* ⇐*analyze migration latency (moves)*) for the federate migrations. Based on this analysis the simulation execution is generated.

The CLB matches the most overloaded with the most underloaded, as shown in Figure 4.2 . As detailed in Algorithm 6, the balancing system requests more information about the topological distance for both overloaded and underloaded resources. The CLB compares the load with the matching that has been assigned, and analyzes this by using the following equation: (*number_fed (src_rsc) & number_fed (dst_rsc) & (min* ∗ *φ) & (min* ∗ *δ)*). The comparison is divided into two analyses. The first analysis is applied to identify the potential candidate that will receive the federate (if *dst_rsc* < *min)* if the load of the destination resource (*dst_rsc*) is less than the minimum load (*min*).

Next, the number of federates in the resources is checked. If the number of federates is one or more than one, then the next comparison is applied to check the computational load. If the computational load is larger than the threshold, this results in the minimum load multiplying the parameter φ (min ∗ φ). The firs threshold analysis represents the load for one process in a CPU queue that is shown by the φ.

Whenever the comparison is correct, then the migration move is generated. However, if the destination resource (*dst_rsc* < *min)* is larger than the minimum load (*min*), then the second analysis is applied. The result of the destination resource subtracting the source resource should be larger than the result of the threshold *(min* ∗ δ). The

second threshold represents the value equal to or less than the load created by the CPU queue. When all of the conditions are correct, then the migration move is generated. Furthermore, when any resources do not pass through this comparison, then the balancing system ignores this match.

Algorithm 6: Pair-Match Evaluation Algorithm [15]

Require*: src_rsc, dst_rsc*
selected federate *select federate smallestLatency(src_rsc)*
if *dst_rsc* < *min* **then**
    if *number fed(src_rsc)* ≥ *1 & src_rsc* > *(min* *φ) then*
        Δt′ *Δt* × *α*
        create migration move(src_rsc, dst_rsc, selected federate)
    else *if number fed(src_rsc)* > *1* **then**
        Δt′ *Δt*
        create migration move(src_rsc, dst_rsc, selected federate)
    end
else if *(dst_rsc – src_rsc)* > *(min* *δ)* **then**
    if *number fed(src_rsc)* ≥ *1 AND (dst_rsc – src_rsc)* > *(min* *φ)* **then**
    Δt′ *Δt* × *α*
        create migration move(src_rsc, dst_rsc, selected federate)
    else *if number fed(src_rsc)* > *1* **then**
        Δt′ *Δt*
        create migration move(src_rsc, dst_rsc, selected federate)
    end
end
send migration moves(estMigTime & estimateMigGain)

As detailed in Algorithm 7, the balancing system requests further information that is necessary to calculate the estimation for the time delay and the time gain. This calculation enables the balancing system to be aware of migration when the redistributed simulation load is analyzed. The balancing system requests the bandwidth usage for the destination resources (*dst_usag),* and this information is provided by the nload tool. The balancing system also requests the current load of source resource (*src_load*) and the current load of destination resource (*dst_load*).

Algorithm 7 Migration Move Algorithm

Require: *dst_usag, avg_usag, src_load, dst_load, Porc_time, link_cap*
while *migrationMove* ≠ ∅ **do**
estMigTime ⇐ *estMigTime(dst_usag, avg_usag, Porc_time, link_cap)*
estimateMigGain ⇐ *estimateMigGain(src_load, dst_load,,Δt')*
Return: *migrationMove, estimatedMigGain, estimatedMigTime*
end

Return: *migrationMove*

The final request is the process time (*Proc_time*). The *Proc_time* is the result of the ($\boldsymbol{Finish_{time}} - \boldsymbol{Arrival_{time}}$). This information is provided by HLA/RTI components. Furthermore, the link bandwidth is fixed and equal to 512 Mega-bits.

The CLB gathers the information that is requested and then starts to estimate the migration delay (*estMigTime*) and time gain (e*stimateMigGain*). As described in formula (4.2*)*, the estimation of the migration delay (t_e) for each federate is calculated with the past process time multiplied by the result of the current usage of the destination resources, divided by like capacity. As described in formula (4.2*)*, the estimation of the time gain (t_d) is calculated with Δt and with the result of the current load of destination resource, divided by the current load source resource, and subtracted by the minimum load. As mentioned above, the comparison is applied between the (t_e) and (t_d). However, the balancing system filters the match that shows that (t_e) is larger than (t_d). Furthermore, the estimation of time delay and time gain is presented in milliseconds for the purpose of comparison.

Algorithm 8: Pair-Match Evaluation Algorithm [15]

Require*: mig_moves*
if *mig_moves* ≠ ∅ **then**
    timeGain ⇐ *calculateGain(mig_moves)*
    expectedMigrationDelay ← *calculate(mig_moves)*
    while *timeGain* ≤ *expectedMigrationDelay* **do**
        mig_moves ⇐ *eliminate smallest gain(mig_moves)*
        timeGain ⇐ *calculateGain(mig_moves)*
        expectedMigrationDelay ⇐ *calculate(mig_moves)*
    end
end
Return*: mig_moves*

The list of the migration move candidates that pass through the filtering conditions is created. As detailed in Algorithm 8, the balancing system calculates the overall sum of migration time gains (*timeGain* ⇐ *calculateGain (mig_moves)*) and the overall estimation of migration delays (*expectedMigrationDelay* ← *calculate (mig_moves)*). The balancing system calculates the overall sum of migration time gains (t_{ds}) by accumulating all of the migration time gains, as detailed in formula (3.3). The balancing system calculates the overall estimation of migration delays (t_{es}) using the largest migration delays in the set of migration candidates to detect which list of matches will provide a high performance gain, as detailed in formulas (3.4) and (4.4).

$$\alpha = \frac{2 * \bar{t_e} - 1 + t_{e_{largest}}}{\left(2 * t_{e_{largest}}\right)} * (k + 1) \qquad \textbf{(4.4)}$$

α: influence of other migration delays on overall estimation
$\bar{t_e}$: mean of migration delay
$t_{e_{largest}}$: largest migration delay
k: number of migrations with delay larger than mean

The balancing system applies a comparison between (t_{ds}) and (t_{es}). Comparisons are repeatedly applied until the (t_{ds}) is shown to be larger than (t_{es}). This comparison eliminates the matches that have a small gain. When the CLB receives the final list of migration moves, then such a migration call is issued from the CLB to the MM through the LLB as shown in Algorithms 4 and 5. The MM starts to migrate federates from the overloaded to the underloaded. When the MM finishes the migration procedure, the balancing system is updated with the information that is used in the next balancing cycle. Furthermore, the inter-domain area is calculated to adjust the *relFactor* for the next balancing cycle (*adjust_factor (relFactor, overloaded RSCs, moves)*).

Chapter 5. Performance Analysis and Discussion

Estimations have been conducted in order to evaluate the performance of the proposed modified and enhanced distributed load-balancing system. We compared our proposed solution with the dynamic balancing system. Different imbalance scenarios have been used in this comparative analysis. As mentioned previously, our goal is to have a dynamic load-balancing system to decrease simulation execution time. After estimating all of the previously discussed sets of rules and procedures, results have shown that the proposed balancing technique method provides an enhancement by decreasing the number of federates which results in decreasing the simulation time. From the results of our analysis, the simulation execution time with different imbalances has been enhanced significantly.

5.1. Environment and Scenario Analysis

We provide a thorough description of our analytical results, which were mentioned in Chapter 4. As shown in Table 5-1, the analyses were calculated in a distributed environment composed of different scenarios. We defined five different imbalance scenarios of our evaluation, as shown in Table 5-1 (a). The imbalance scenarios have been used to evaluate various nodes and federates, as shown in Table 5-1 (b).

Table 5-1 General Parameters for the Evaluation

(a) Imbalance Scenarios	
Imbalance Statue	**Ability to Work**
Low Load Stress	75%
Low-medium Load Stress	67%
Medium Load Stress	50%
High-medium Load Stress	33%
High Load Stress	25%

(b) Number of Nodes and Federates	
Number of Nodes	**Number of Federates**
10	1,10,25,50, 75,100,150, 250,350,500, 700,850,900, and 1000
50	
100	
250	
500	
750	
1000	

As shown in Table 5-1 (a), we propose dividing the imbalance scenarios into five levels ranging from 25 to 75%. Also, we propose a range of numbers of nodes and federates to observe the balancing behavior in different situations, as shown in Table 5-1 (b). In order to realize the analysis, we divided the migration delay into four levels, as shown in Table 5-2. Each estimation consists of 1 to 1000 federates; furthermore, the estimation has been calculated on each imbalance scenarios that has been applied on each number of nodes and with the entire migration delay configuration.

Table 5-2 Migration Delay Systems

Low Migration delay	Mid Migration delay (A)	Mid Migration delay (B)	High Migration delay
1 second	2 seconds	3 seconds	5 seconds

5.2. Evaluation of the Proposed Balancing Scheme

In this balancing evaluation, analyses have been conducted to evaluate the performance of the proposed enhancement of the balancing system based on the migration-aware scheme by observing the balancing efficiency gain that can improve the simulation performance.

In the analysis, the proposed dynamic balancing scheme was compared with four estimation schemes and more specifically with the dynamic balancing system. The baseline was obtained by the following five estimated calculations. First, the simulation federates were evenly distributed on the available resources and the simulation time was estimated without any balancing - referred to in the graphs as Sim_time_no_imbl. Second, the simulation federates were distributed on a reduced number of available resources according to the imbalance scenarios, and the simulation time was computed without considering any balancing - referred to in the graphs as Sim_time_Imbl. Third, the simulation federates were distributed on a reduced number of available resources according to the imbalance scenarios, and the simulation time was calculated by using the dynamic balancing system - referred to in the graphs as Sim_time_balan. The balancing system modified the load distribution as needed. Finally, the last the simulation federates were distributed on a reduced number of available resources according to the imbalance scenarios, and the simulation time was estimated by using our proposed enhancement based on migration-aware balancing system - referred to in the graphs as Sim_time_Flex_Δt.

By comparing balancing efficiency of the dynamic balancing system with our enhancement balancing technique, a reduction in simulation time has been observed. However,

this improvement is better understood through the number of migrations that were performed by the balancing system. Consequently, in the graphs that show performance gain, the Y-axis represents the execution time of the simulations. In the graphs that show balancing effectiveness, Y-axis represents the number of migrations. However, the best estimation (*Sim_time_no_imbl*) and worst estimation (*Sim_time_imbl*) were not compared, except when unusual behaviour occurred

As we described earlier, there are many analytical results based on imbalance scenarios, number of nodes, and the migration delay system. Here we will describe two results in each imbalance scenario by using a different migration delay system and number of nodes. In addition, a Table 5-3 is provided at the end of this chapter to summarize the results. As described in Table 5-1 (a), the results are discussed according to the five load stress categories.

5.2.1 Evaluations with Low Load Stress

In the following graphs, we have chosen two results that represent the Low Load Stress, as shown in Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4. According to the graph in Figure 5.1, the four balancing schemes are able to present different performance gains. However, the execution time with the distributed balancing system has shown a slight improvement.

As described in Figure 5.2, the number of migrations increases significantly with 150 migration federates; therefore, the performance is not satisfactory due to the migration overhead.

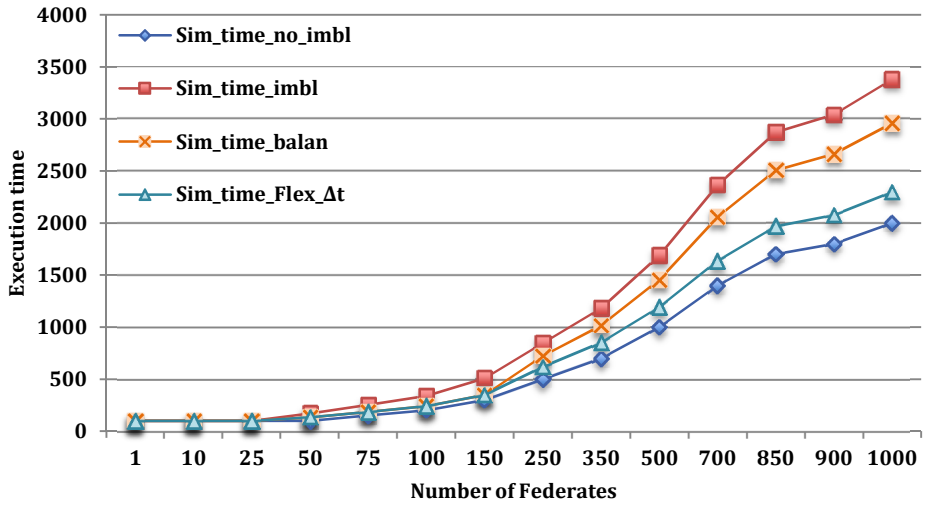


Figure 5.1 Comparison of Performance Analysis for an Increasing Number of Federates on 50 nodes with Low Load Stress and Mid Migration Delay (B)

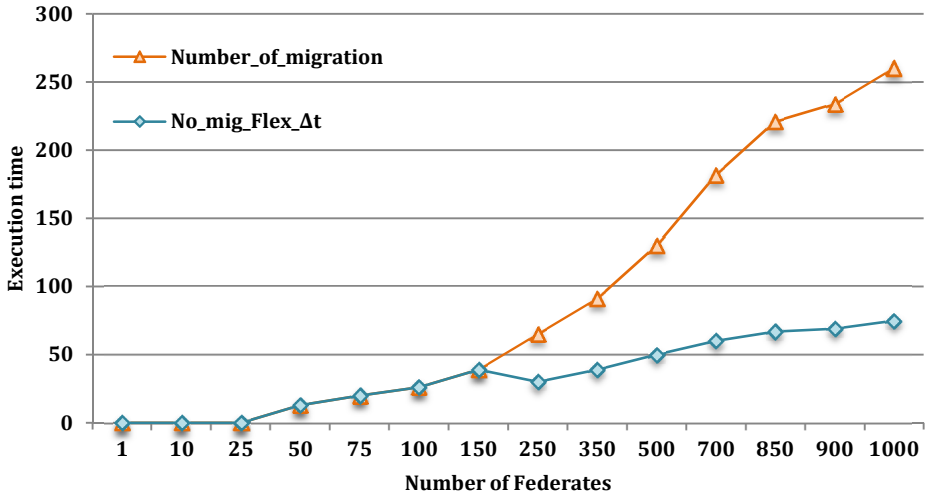


Figure 5.2 Comparison between Number of Migrations for an Increasing Number of Federates on 50 nodes with Low Load Stress and Mid Migration Delay (B)

From the results of our proposed balancing and dynamic balancing systems, the execution time of the simulation has been enhanced up to 20% and the number of migrations has decreased about 50%.

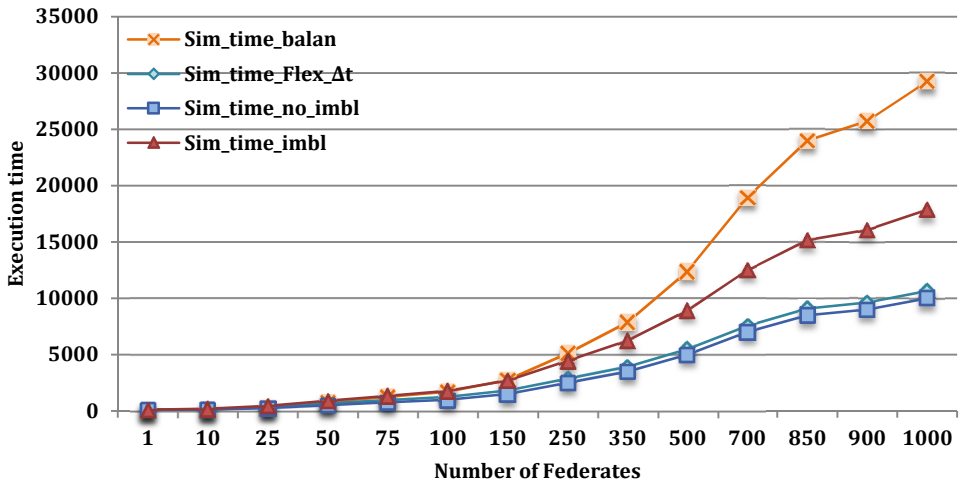


Figure 5.3 Comparison of Performance Analysis for an Increasing Number of Federates on 10 nodes with Low Load Stress and High Migration Delay

As shown in Figure 5.3, the distributed balancing approach shows a worse simulation time. The dynamic balancing approach reacted to an uneven performance gain due to the large amount of federate migrations. However, compared to the dynamic balancing approach, the execution time has been improved by about 30% when system has reached 150 federates. Furthermore, the results in the same graph show a 34% decrease in the simulation time in comparison to the simulation time that was executed without the dynamic balancing system.

As shown in Figure 5.4, the number of migrations in dynamic balancing approach has significantly increased, but in the proposed balancing technique migration has decreased up to 45% compared to the number of the migrations dynamic balancing approach. The simulation execution time on the dynamic balancing approach has increased due to the high migration delay.

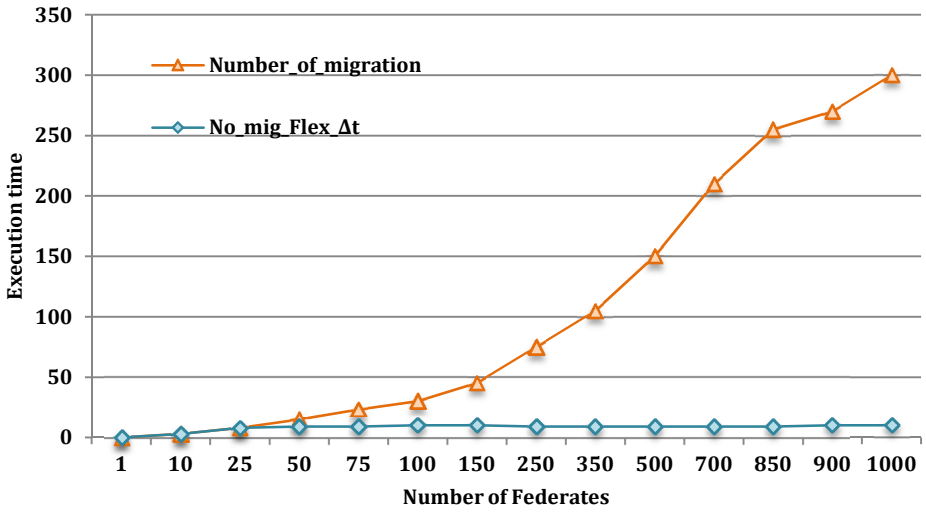


Figure 5.4 Comparison between Number of Migrations for an Increasing Number of Federates on 10 nodes with Low Load Stress and High Migration Delay

From the results of our analyses in Figure 5.1 and Figure 5.3, our proposed technique with low load stress has been decreased the execution time between 20-35% in comparison to the dynamic approach.

5.2.2 Evaluations with Low-medium Load Stress

As depicted in Figure 5.5, our proposed and dynamic balancing techniques grew at a similar rate until the dynamic balancing technique reached 250 federates and there was a slight change. However, our proposed technique decreased the simulation execution time up to 10%.

For the same evaluation, Figure 5.6 shows the number of migrations required for each balancing system to provide an improved simulation execution time. However, the proposed technique decreased the number of migrations between 20 -25% in comparison to the dynamic balancing technique.

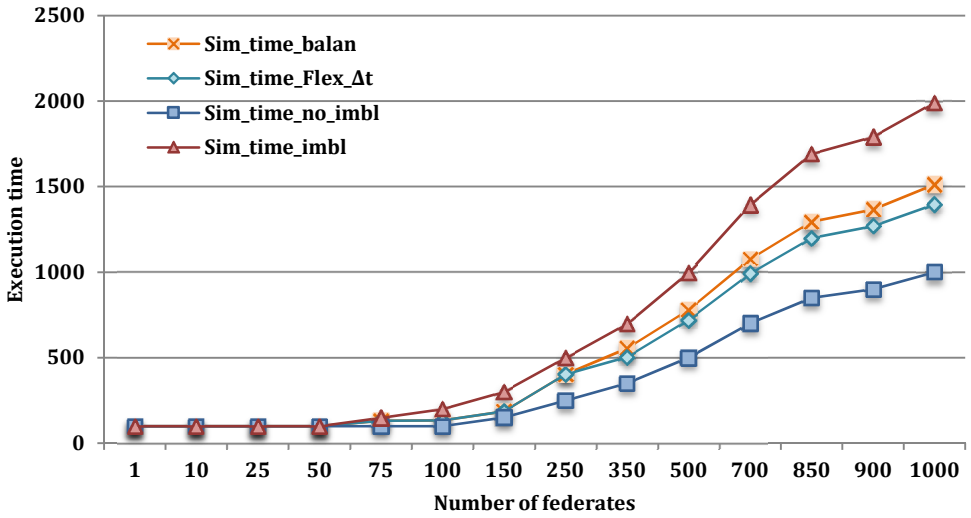


Figure 5.5 Comparison of Performance Analysis for an Increasing Number of Federates on 100 Nodes with Low-medium Load Stress and Low Migration Delay

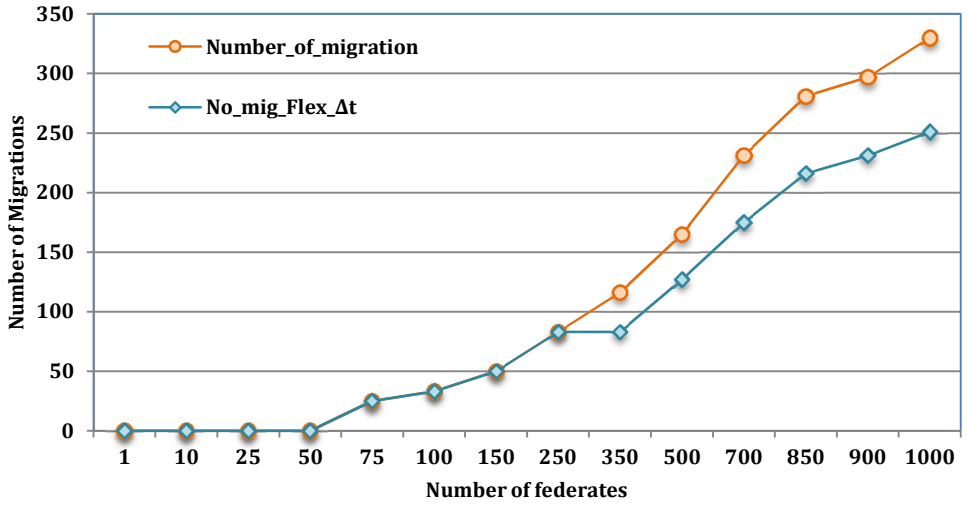


Figure 5.6 Comparison between Number of Migrations for an Increasing Number of Federates on 100 Nodes with Low-medium Load Stress and Low Migration Delay

As presented in Figure 5.7, the proposed balancing technique was more effective in reducing the simulation execution time, which was improved by up to 15%. While the dynamic, and proposed balancing techniques reacted similarly, when the simulation reached 350 federates, the proposed technique reacted differently.

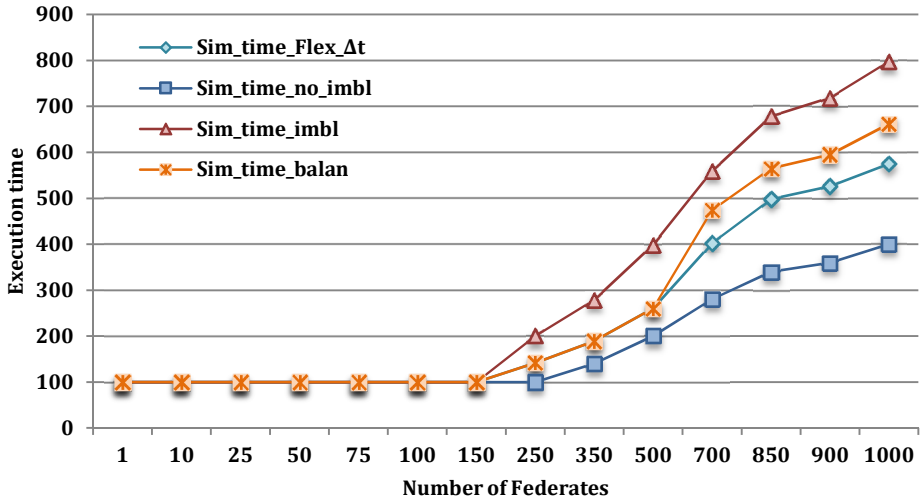


Figure 5.7 Comparison of Performance Analysis for an Increasing Number of Federates on 250 Nodes with Low-medium Load Stress and High Migration Delay

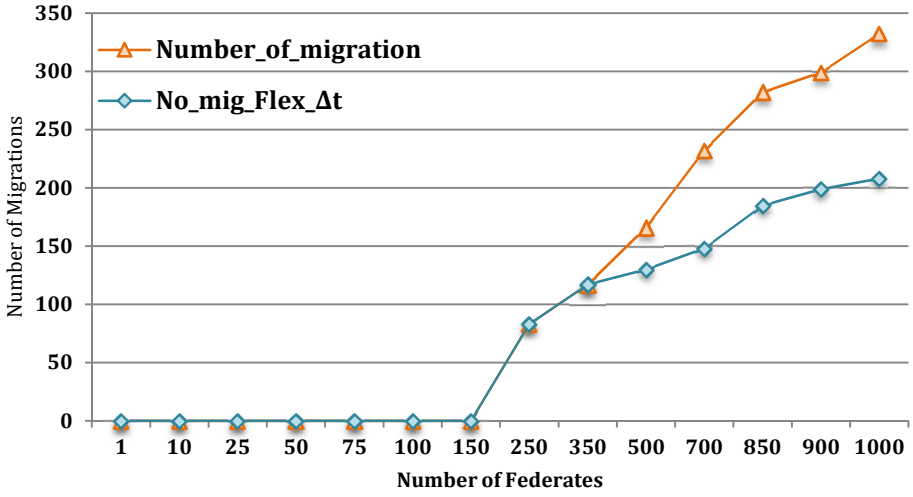


Figure 5.8 Comparison between Number of Migrations for an Increasing Number of Federates on 250 Nodes with Low-medium Load Stress and High Migration Delay

As shown in Figure 5.8, the number of migrations increased when the simulation reached 150 federates. Furthermore, the dynamic and proposed techniques increased the number of migrations similarly, but once the simulation execution reached 117 federates, the proposed technique was able to decrease the number of migrations up to 30%, which is a significant improvement. The proposed technique provides the best performance with a 10 - 15% range of improvements in comparison to the dynamic balancing technique.

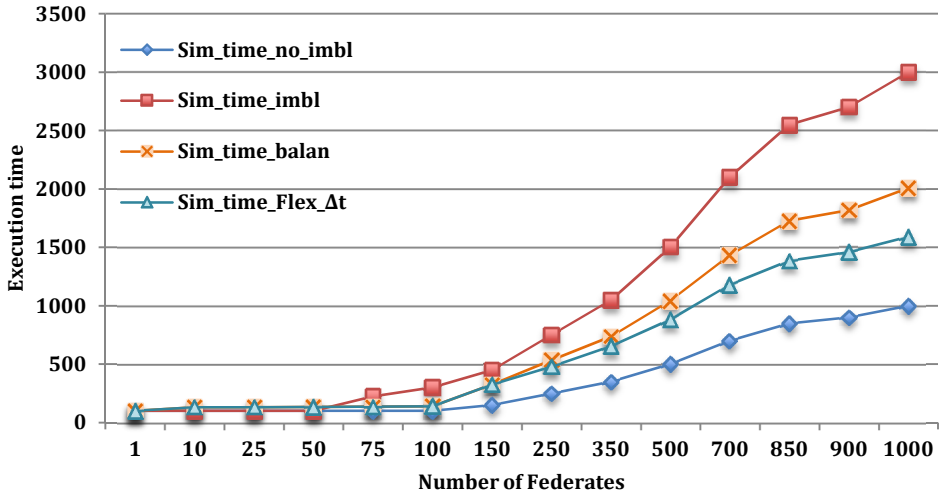


Figure 5.9 Comparison of Performance Analysis for an Increasing Number of Federates on 100 Nodes with Medium Load Stress and Mid Migration Delay (A)

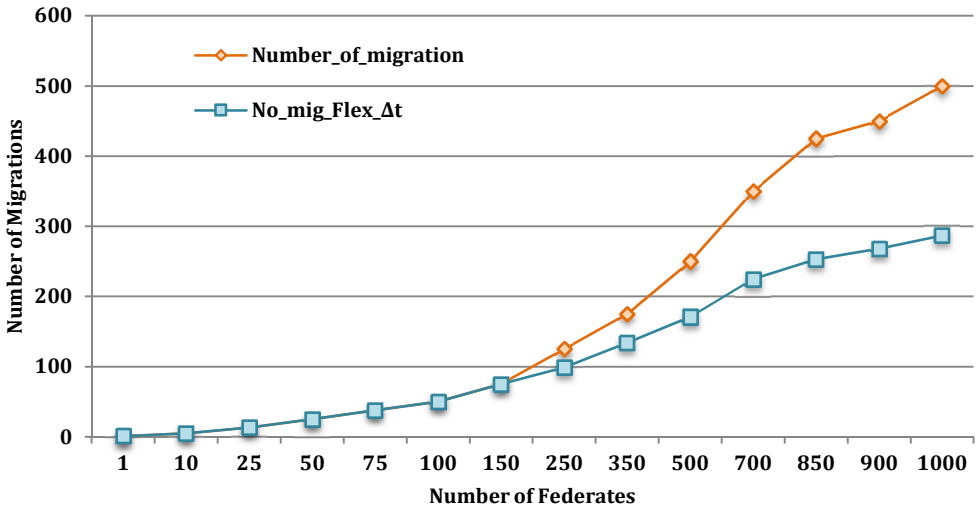


Figure 5.10 Comparison between Number of Migrations for an Increasing Number of Federates on 100 Nodes with Medium Load Stress and Mid Migration Delay (A)

5.2.3Evaluations with Medium Load Stress

According to the graph in Figure 5.9, the balancing scheme reacted by redistributing the load with a good performance gain, but the dynamic balancing system was able to decrease the simulation execution time up to 10%. However, the proposed balancing

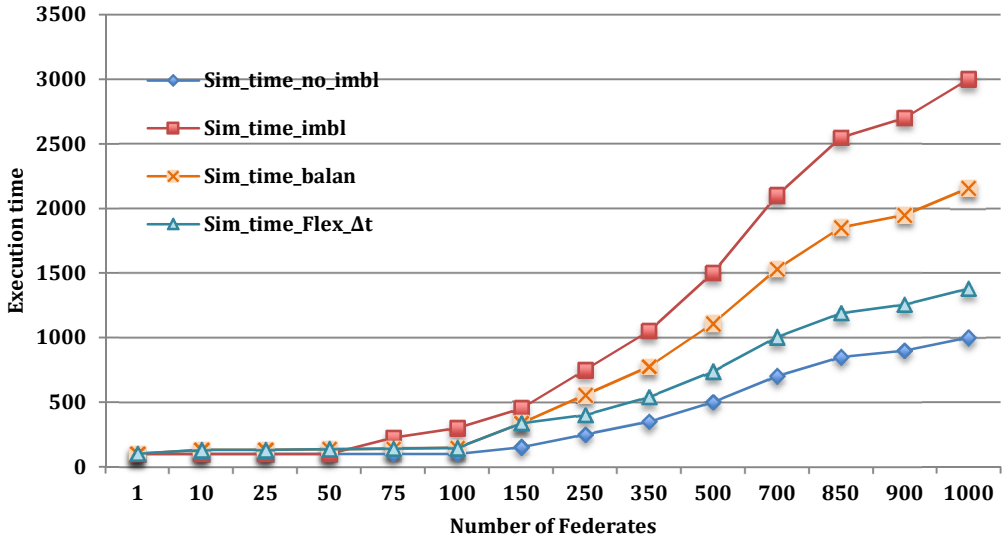


Figure 5.11 Comparison of Performance Analysis for an Increasing Number of Federates on 100 Nodes with Medium Load Stress and Mid Migration Delay (B)

system had up to 20% improved achievement in comparison to the performance gain of the dynamic balancing system.

When observing the number of migrations in Figure 5.10, the dynamic balancing system and previous balancing systems has decreased the same number of migrations. However, a difference occurred when the simulation reached 150 federates after which point the dynamic balancing system decreased the number of migrations up to 16%. The proposed balancing system decreased the number of migrations up to 20% in comparison to the number of migrations of the previous balancing system, which shows better performance gain, as shown in Figure 5.9.

As described in Figure 5.11, the four balancing schemes were able to present different performance gains. When comparing the results of our scheme to the dynamic balancing scheme, our scheme presented an improvement for the balancing redistribution,

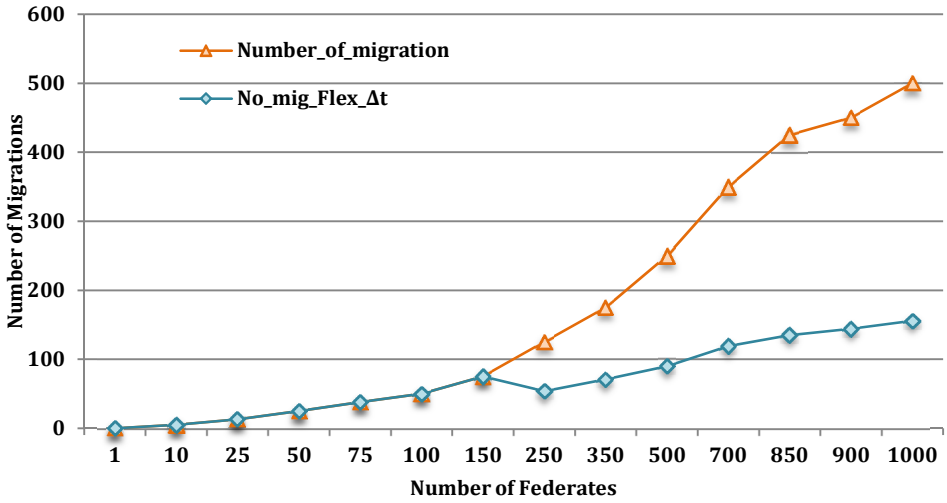


Figure 5.12 Comparison between Number of Migrations for an Increasing Number of Federates on 100 Nodes with Medium Load Stress and Mid Migration Delay (B)

and the performance gain improved up to 45%. Our scheme produced more efficient results by reducing the simulation execution time with mid-migration delay (B) imbalances due to the number of migration that has been decreased.

As shown in Figure 5.12, our scheme was able to decrease the number of migrations, but it offered only a good performance gain. However, the dynamic balancing and proposed techniques increased the number of migrations similarly, but once the simulation execution reached 150 federates, our technique was able to decrease the number of migrations that was considered on the dynamic balancing approach as migration federates, which resulted in increasing the simulation execution time due to the large amount of federate migrations. Thus, the curve fairly represents the difference between the dynamic and our proposed balancing schemes. Our balancing scheme decreased the number of migrations between 40 – 50 % in comparison to the dynamic scheme, which decreased the number of migrations up to 15%.

From the results in Figure 5.9 and Figure 5.11, the simulation execution time with

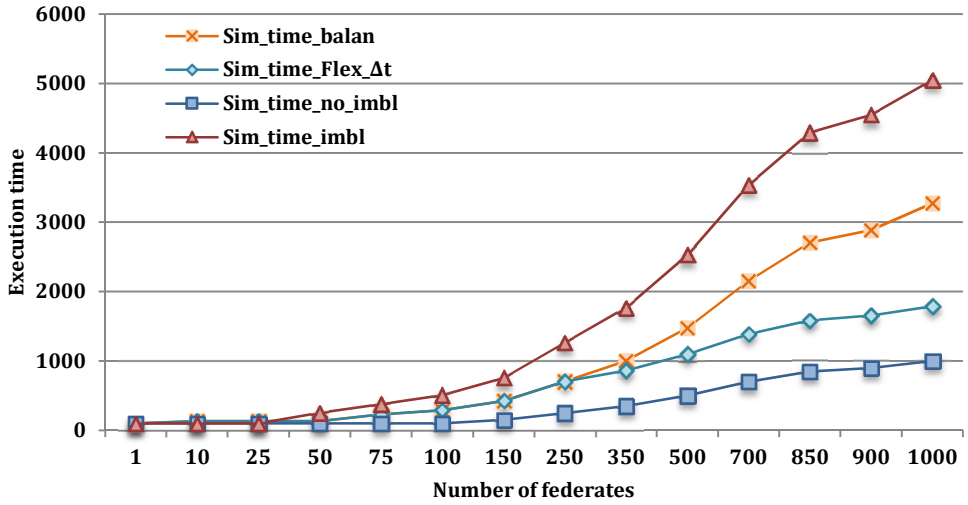


Figure 5.13 Comparison of Performance Analysis for an Increasing Number of Federates on 100 Nodes with High-medium Load Stress and Low Migration Delay

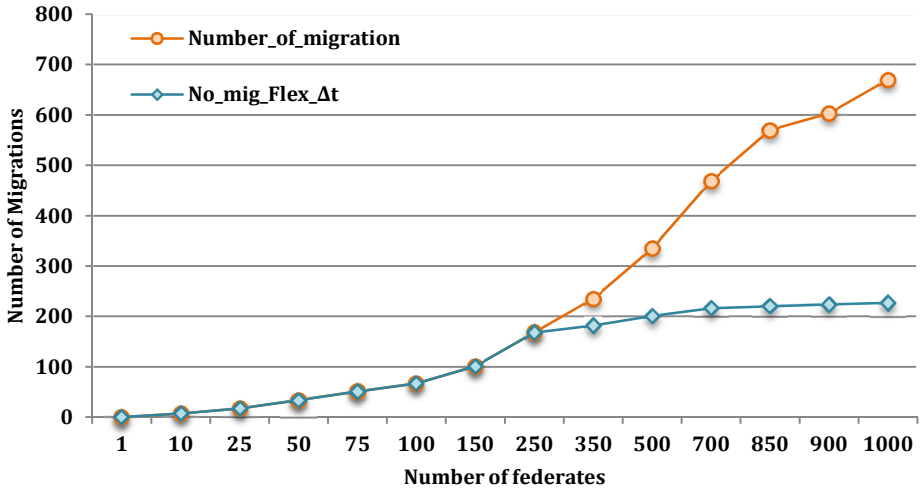


Figure 5.14 Comparison between Number of Migrations for an Increasing Number of Federates on 100 Nodes with High-medium Load Stress and Low Migration Delay

medium load stress has been enhanced between 20 - 40% with the proposed balancing technique in comparison to the dynamic balancing system.

5.2.4 Evaluations with High-medium Load Stress

As shown in Figure 5.13 , in comparison to the performance gain between the dynamic balancing technique and our proposed technique, the proposed technique decreased the simulation execution time up to 45% in comparison to the dynamic balancing

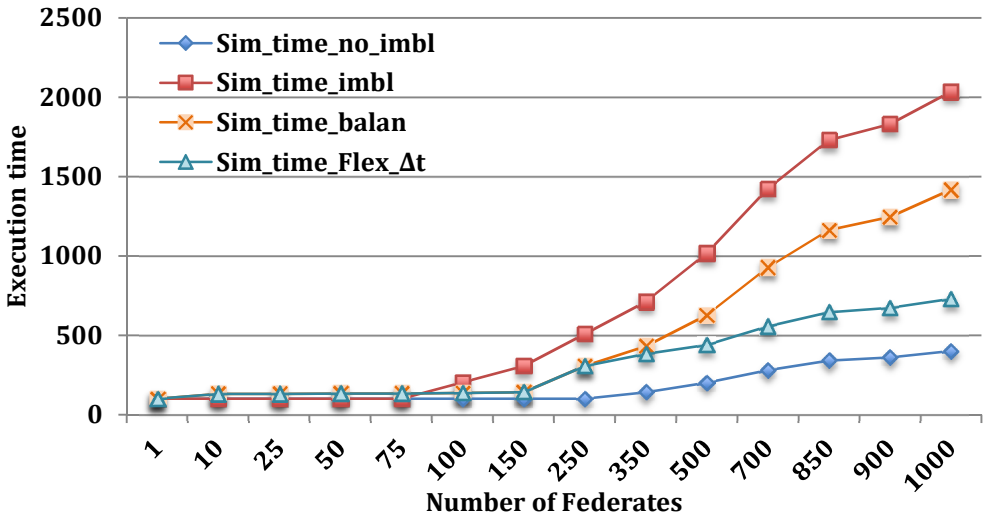


Figure 5.15 Comparison of Performance Analysis for an Increasing Number of Federates on 250 nodes with High-medium Load Stress and Mid Migration Delay (B)

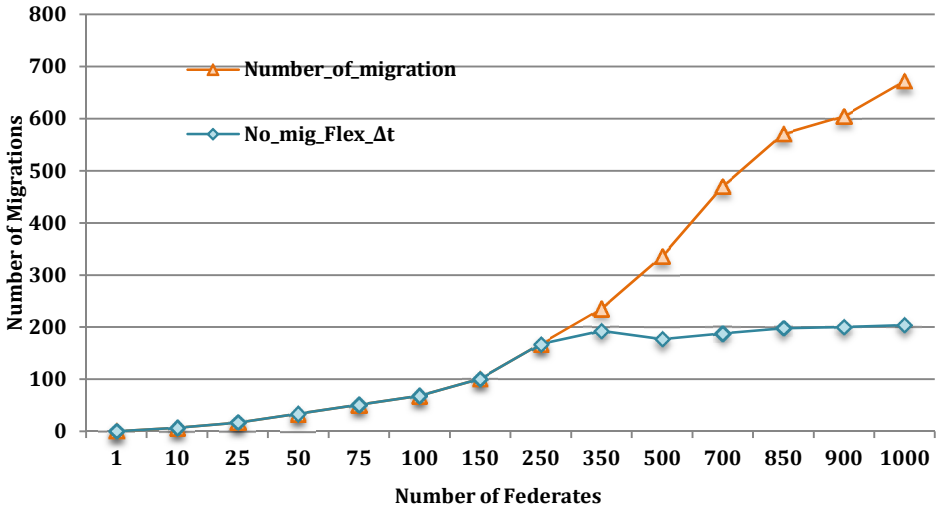


Figure 5.16 Comparison between Number of Migrations for an Increasing Number of Federates on 250 nodes with High-medium Load Stress and Mid Migration Delay (B)

technique, because of the large number of migrations that have been considered in the dynamic load balancing technique.

For the same analysis, the results in Figure 5.14 show the large number of migrations for the dynamic technique, which influenced the final performance gain result. The dynamic technique's overhead was almost 48% more than the proposed technique migration.

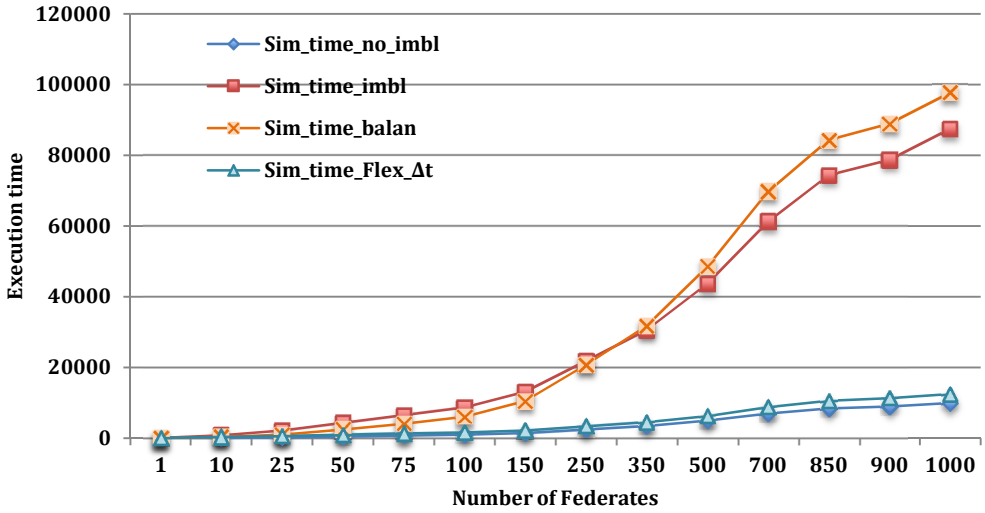


Figure 5.17 Comparison of Performance Analysis for an Increasing Number of Federates on 10 Nodes with High Load Stress and Mid Migration Delay (B)

As shown in the inset graph in Figure 5.15 and Figure 5.16, our proposed approach demonstrated a performance improvement between 45 – 55 % in comparison to the dynamic approach for some cases.

In all cases our approach provided the best performance with a 25 - 35% range of improvement in comparison to the dynamic approach.

5.2.5 Evaluations with High Load Stress

As depicted in Figure 5.17, the dynamic balancing scheme redistributed the load and showed a slight performance gain, but when the dynamic balancing system reached 350 federates, a worse simulation execution time was shown in comparison to the our schemes. Moreover, the simulation execution time was increased by more than 20%. However, the proposed balancing scheme achieved up to 65% improvement in comparison to the dynamic balancing scheme performance gain. Consequently, the proposed balancing scheme was more effective in reducing the simulation execution time as shown in Figure 5.18.

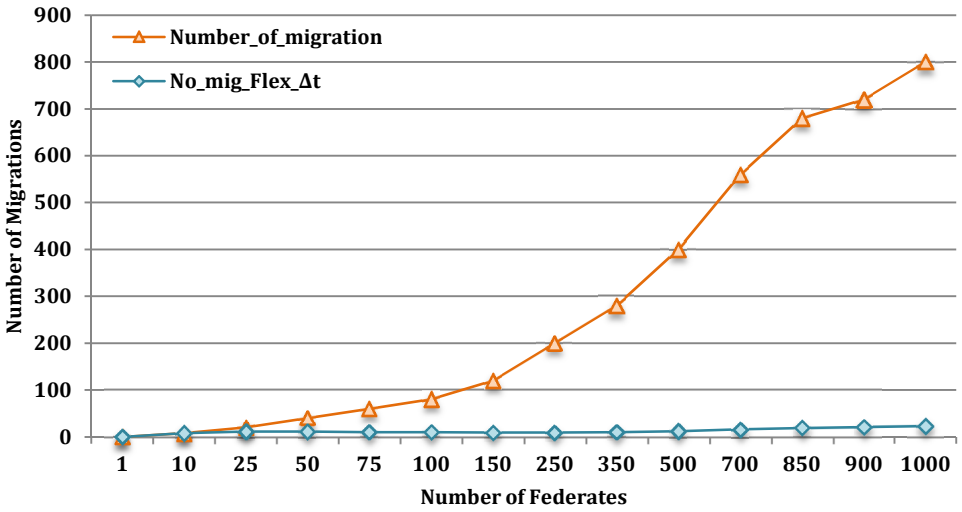


Figure 5.18 Comparison between Number of Migrations for an Increasing Number of Federates on 10 Nodes with High Load Stress and Mid Migration Delay (B)

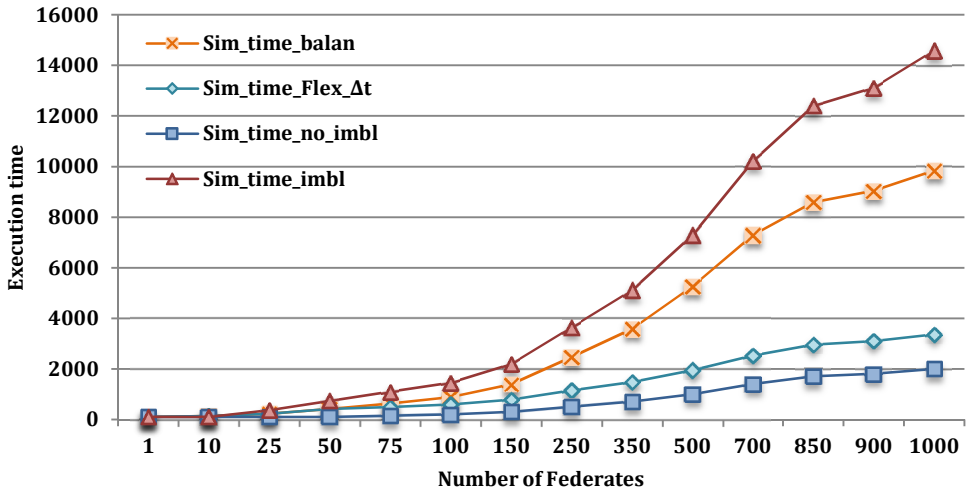


Figure 5.19 Comparison of Performance Analysis for an Increasing Number of Federates on 50 nodes with High Load Stress and High Migration Delay

According to the curves, both the proposed scheme has been developed the balancing efficiency, since the proposed technique reduced the amount of migrations to achieve better performance gains. However, in comparing between the proposed and the dynamic balancing schemes, the proposed balancing scheme was able to decrease the number of migrations up to 65%, since the number of migrations was reduced to achieve a better performance gain.

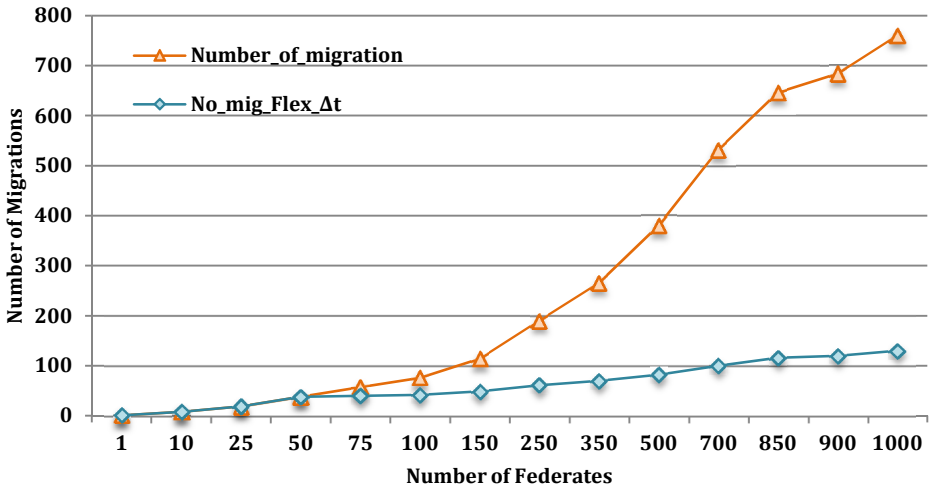


Figure 5.20 Comparison between Number of Migrations for an Increasing Number of Federates on 50 nodes with High Load Stress and High Migration Delay

As shown in Figure 5.19, the dynamic balancing system and our balancing technique reacted similarly, but our proposed balancing technique showed a better enhancement on the execution time and it decreased the execution time almost 60% in comparison to the dynamic balancing technique. However, the dynamic balancing system showed a worse simulation execution time due to the overhead, which was increased by more than 55% in comparison to the performance of our proposed balancing system.

As shown in Figure 5.20, the dynamic balancing system has a large number of migrations, which influenced the final performance gain result. In comparing our proposed balancing system with the dynamic balancing system, the performance of our proposed system was enhanced between 55 - 65% by decreasing the number of migrations required by our balancing scheme to obtain a simulation performance improvement.

By observing the inset graphs in Figure 5.17 and Figure 5.19, the proposed schemes show an improved performance ranging from 50 - 65% in comparison to the dynamic scheme for some cases.

5.3. Summary

As described previously, the proposed and dynamic balancing techniques have presented a variety of performance gains. However, the proposed balancing technique showed better performance than the performance of the dynamic balancing system. The most significant benefit was that the Δt was allowed to be flexible, when it had previously been considered fixed.

As presented in Table 5-3, the proposed balancing technique shows good results whenever the number of federates is close to double the number of nodes. Moreover, a better behavior is shown when the migration delay is in high load stress. This is because the proposed balancing system was able to obtain a useful number of migrations that shows a better performance gain. However, the proposed balancing technique reacted similarly with no differences between the proposed balancing technique whenever the number of nodes is close to or higher than the number of federates. Furthermore, the proposed balancing scheme and the dynamic balancing system did not redistribute the load when the simulations did not require any load balancing.

Table 5-3 Detailed Comparison of Performance between Proposed and Previous Balancing System with the Distributed Balancing System

	Low	Mid (A)	Mid (B)	High	Avg.	Diff	No Nodes
Low Load Stress with 1 – 1000 Federates							
Our Technique	27.5%	37.6%	49.6%	58.3%	43.2%	2.0%	10
Previous Technique	26.5%	36.2%	46.8%	55.2%	41.2%		
Our Technique	5.0%	8.8%	18.8%	20.5%	13.3%	2.0%	50
Previous Technique	1.9%	6.6%	17.3%	19.3%	11.3%		
Our Technique	0.2%	1.1%	9.7%	8.7%	4.9%	1.7%	100
Previous Technique	0.0%	0.0%	6.5%	6.2%	3.2%		
Our Technique	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	250
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
High-medium Load Stress with 1 – 1000 Federates							
Our Technique	44.0%	56.5%	65.4%	73.0%	59.7%	1.1%	10
Previous Technique	42.7%	54.6%	65.0%	72.5%	58.7%		
Our Technique	16.1%	22.7%	34.9%	39.8%	28.4%	1.8%	50
Previous Technique	13.4%	21.1%	33.7%	38.2%	26.6%		
Our Technique	6.4%	10.1%	22.2%	24.0%	15.7%	3.6%	100
Previous Technique	0.4%	6.4%	20.0%	21.5%	12.0%		
Our Technique	0.0%	0.4%	7.4%	6.2%	3.5%	3.5%	250
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
Our Technique	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	500
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
Medium Load Stress with 1 – 1000 Federates							
Our Technique	57.4%	68.9%	77.2%	83.0%	71.6%	0.5%	10
Previous Technique	56.8%	68.3%	76.8%	82.6%	71.2%		
Our Technique	24.2%	33.3%	46.1%	51.3%	38.7%	2.5%	50
Previous Technique	20.5%	30.2%	44.6%	49.6%	36.3%		
Our Technique	8.9%	16.1%	31.0%	33.3%	22.3%	5.0%	100
Previous Technique	1.6%	9.8%	27.7%	30.2%	17.3%		
Our Technique	0.0%	0.0%	10.7%	9.3%	5.0%	5.0%	250
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
Our Technique	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	500
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		

High-medium Load Stress with 1 – 1000 Federates							
Our Technique	74.5%	82.6%	88.3%	91.4%	84.2%	0.5%	10
Previous Technique	73.8%	82.2%	88.0%	91.0%	83.7%		
Our Technique	48.9%	59.2%	71.4%	76.1%	63.9%	1.5%	50
Previous Technique	46.7%	57.7%	70.2%	74.9%	62.4%		
Our Technique	32.7%	42.8%	58.4%	62.9%	49.2%	2.7%	100
Previous Technique	28.9%	40.1%	56.4%	60.8%	46.6%		
Our Technique	8.5%	15.2%	34.0%	35.3%	23.2%	23.2%	250
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
Our Technique	0.0%	1.0%	13.7%	13.1%	7.0%	7.0%	500
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
Our Technique	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	750
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
High Load Stress with 1 – 1000 Federates							
Our Technique	68.2%	77.3%	86.1%	88.2%	79.9%	1.2%	10
Previous Technique	67.4%	76.6%	83.3%	87.7%	78.7%		
Our Technique	29.5%	40.6%	53.1%	61.3%	46.2%	2.4%	50
Previous Technique	26.2%	37.7%	52.1%	59.2%	43.8%		
Our Technique	7.2%	17.9%	32.2%	41.9%	24.8%	3.9%	100
Previous Technique	2.0%	13.0%	30.6%	38.0%	20.9%		
Our Technique	0.0%	0.0%	4.2%	11.0%	3.8%	3.8%	250
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		
Our Technique	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	500
Previous Technique	0.0%	0.0%	0.0%	0.0%	0.0%		

Chapter 6. Conclusions and Future Work

Our proposed balancing scheme has produced a minimum required number of migrations avoiding costly latencies, which shows an enhancement over the previous load redistribution system. The proposed balancing system has shown that the execution time has been decreased, which is a strong indicator of improvement in the simulation performance. Moreover, each migration analysis procedure was evaluated based on the time delay and time gain that were calculated according to the matches between underloaded and overloaded candidates. A comparison was made between time delay and time gain. Basically, whenever the time gain was larger than the time delay, this match was considered a candidate for migration. Also, an analysis was performed on the overall sum of time delays and time gains. Consequently, this pair-matching mechanism prevented the balancing system to decide upon performing costly migrations to adapt load during the run-time of distributed simulations. Our studies indeed verified that a flexible Δt showed promising results, benefiting the balancing efficiency of the base migration-aware balancing scheme. In this conclusion, we summarize our contributions and provide some useful directions for future work in this subject.

6.1. Summary of Contributions

As we have described previously, the contributions of this work can be summarized on the number of enhancements introduce onto the migration-aware load balancing system. In

order to achieve such increase in load management efficiency, we generated the following improvements in the load analysis and decision-making scheme:

1- We proposed new estimation metrics for the migration delay and time gain.
    a. Δt is estimated in a flexible fashion, as calculated based on real time migration.
    b. Estimation of time gain is enhanced, as calculated based on previous computed values and a flexible Δt.
    c. Estimation of migration delay is calculated by using new metrics.
2- We developed more restrictive and precise decision-making for the creation of migration federates.
3- Our enhancement solution has effectively avoided costly migrations and consequently decreased execution time.

6.2. Future Work

In our future work, we plan to implement our proposed federate migration scheme on the PARADISE Laboratory's clusters at the University of Ottawa. This will allow observation and monitoring of the balancing behavior of our proposed scheme. Furthermore, we are planning to study further with oscillating migration delay. We will use different heuristics for defining load imbalances and response to them, such as study on the migration delay awareness in other balancing method for ex: bio-inspired techniques.

Bibliography

[1] A. Boukerche, S. K. Das, and A. Fabbri, "Swimnet: A scalable parallel simulation testbed for wireless and mobile networks," *Wireless Networks,* vol. 7, pp. 467 - 486 , 2001.

[2] P. Zhang and G. Gong, "Atmosphere-affected flight simulation system," *In Proceedings of the 2009 Spring Simulation Multiconference,* pp. 431 - 437 , 2009.

[3] J. Aronson, V. Manikonda, W. Peng, R. Levy, and K. Roth, "An HLA compliant agent based fast-time simulation architecture for analysis of civil aviation concepts," *In Proceedings of the Simulation Interoperability Standards Organization Spring Simulation Interoperability Workshop,* 2003.

[4] P. Wu, G. Cai, D. Zhang, Z. Zhou, and Y. Chen, "HLA-based multi-aircraft combat simulation system," *In Proceedings of the 2nd International Asia Conference on Informatics in Control Automation and Robotics,* vol. 3, pp. 331 - 334 , 2010.

[5] R. E. De Grande and A. Boukerche, "Predictive dynamic load balancing for large-scale HLA-based simulations," *In Proceedings of the IEEE/ACM 15th International Symposium in Distributed Simulation and Real Time Applications,* pp. 4 - 11, 2011.

[6] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed system integration," *Computer,* vol. 35 , no. 6, pp. 37 - 46 , 2002.

[7] IEEE Standard, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Framework and Rules," 1516 2000, September 2000.

[8] IEEE Standard, "Draft Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Object Model Template (OMT) Specification," April. 2000.

[9] IEEE Standard, "Draft Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-Federatiou Interface Specification," April. 2000.

[10] A. Boukerche and R. E. De Grande, "Dynamic load balancing using grid services for HLA-based simulations on large-scale distributed systems," *In Proceedings of the Distributed Simulation and Real Time Applications,* pp. 176 - 183 , 2009.

[11] E. E. Ajaltouni, A. Boukerche, and M. Zhang, "An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure," *In Proceedings of the Distributed Simulation and Real Time Applications,* pp. 61 - 68 , 2008.

[12] D. W. Glazer and C. Tropper, "On process migration and load balancing in time warp," *IEEE Transactions on Parallel and Distributed Systems,* vol. 4, no. 3, pp. 318 - 327 , 1993.

[13] R. E. De Grande, A. Boukerche and H. M. S. Ramadan, "Measuring communication delay for dynamic balancing strategies of distributed virtual simulations," *IEEE Transactions on Instrumentation and Measurement,* vol. 60 (11), pp. 3559 - 3569, 2011.

[14] M. R. Garey and D. S. Johnson, "Computers and Intractability; A Guide to the Theory of NP-Completeness," in *W. H. Freeman & Co*, 1990.

[15] R. E. De Grande, M. A. Almulla, A. Boukerche;, "Measuring and Analyzing Migration Delay for the Computational Load Balancing of Distributed Virtual Simuations," *IEEE Transactions on Instrumentation and Measurement,* vol. 61, no. 12, pp. 3158 - 3174, 2012.

[16] R. E. De Grande and A. Boukerche, "Dynamic load redistribution based on migration latency analysis for distributed virtual simulations," *In Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and Games,* pp. 88 - 93 , 2011.

[17] L. Ke, S. Xiaojun, N. D. Georganas, and A. Boukerche, "SimSITE: The HLA/RTI Based Emergency Preparedness and Response Training Simulation," *In Proceedings of the 11th IEEE International Symposium Distributed Simulation and Real-Time Applications,* pp. 59 - 63 , 2007.

[18] A. Boukerche, and K. Lu, "A novel approach to real-time RTI based distributed simulation system," *In Proceedings of th 38th Annual Simulation Symposium,* pp. 267 - 274 , 2005.

[19] A. Boukerche, N. J. McGraw, C. McGraw, and L. Kaiyuan, "Grid-filtered region-based data distribution management in large-scale distributed simulation systems," *In Proceedings of th 38th Annual Simulation Symposium,* pp. 259 - 266 , 2005.

[20] R. E. De Grande (2012), Dynamic Load Balancing Schemes for Large-Scale HLA-based Simulations, Ph.D. degree in Computer Science, University of Ottawa, Canada.

[21] A. Boukerche and L. Kaiyuan, "Optimized dynamic grid-based DDM protocol for large-scale distributed simulation systems," *In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium,* pp. 6 - 12, 2005.

[22] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scal- able virtual organizations," *International Journal of High Performance Computing Applications,* vol. 15, no. 3, pp. 200 - 222, 2001.

[23] A. Boukerche, A. Roy, and N. Thomas, "Dynamic grid-based multicast group assignment in data distribution management," *In Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications,* pp. 47 - 54 , 2000.

[24] J. Nabrzyski, J. M. Schopf, and J. Weglarz, "Grid Resource Management: State of the Art and Future Trend", Norwell: Kluwer Academic Publishers, 2004.

[25] Globus, *University of Chicago. https://www.globus.org/,* 2008.

[26] A. Boukerche and C. Dzermajko, "Performance evaluation of Data Distribution Management strategies," *Concurrency and Computation: Practice and Experience,* vol. 16, no. 15, pp. 1545 - 1573, 2004.

[27] S. Zhu, Z. Du, and X. Chai, "GDSA: A grid-based distributed simulation architectture," *In Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid,* p. 66, 2006.

[28] K. Rycerz, M. Bubak, M. Malawski, and P. Sloot, "Support for effective and fault tolerant execution of HLA-based applications in the OGSA framework," *In*

Proceedings of the 4th International Conference Computational Science, vol. 3038, pp. 848 - 855 , 2004.

[29] J. Lu¨thi and S. Groβmann., "FT-RSS: A flexible framework for fault tolerant HLA federations," *In Proceedings of the 4th International Conference Computational Science,* pp. 865 - 872 , 2004.

[30] J. Grossmann and S. Luthi, "The resource sharing system: dynamic federate mapping for HLA-based distributed simulation," *In Proceedings of the 15th Workshop on Parallel and Distributed Simulation,* pp. 91 - 98 , 2001.

[31] K. Zajac, M. Bubak, M. Malawski, and P. M. A. Sloot, "Execution and migration management of HLA-based interactive simulations on the grid," *In Proceedings of the fifth international conference on parallel processing and applied mathematics,* pp. 872 - 879 , 2003.

[32] K. Rycerz, M. Bubak, M. Malawski, and P. M. A. Sloot, "A grid service for management of multiple hla federate processes," *In Proceedings of the sixth international conference on parallel processing and applied mathematics,* pp. 699 - 709 , 2005.

[33] K. Zajac, M. Bubak, M. Malawski, and P. Sloot, "Towards a grid management system for HLA-based interactive simulations," *In Proceedings of the 7th International Symposium on Distributed Simulation and Real-Time Applications,* pp. 4 - 11 , 2003.

[34] W. Cai, S. J. Turner, and H. Zhao, "A load management system for running HLA-based distributed simulations over the grid," *In Proceedings of the Sixth IEEE*

International Workshop on Distributed Simulation and Real-Time Applications, pp. 7 - 14, 2002.

[35] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *In Proceedings of the International Conference on Network and Parallel Computing,* pp. 2 - 13 , 2005.

[36] I. Foster and C. Kesselman, "The globus project: A status report," *In Proceedings of the IPPS/SPDP '98 Heterogeneous Computing Workshop,* pp. 4 - 18 , 1998.

[37] Z. Yuan, W. Cai, and M. Y. H. Low, "A framework for executing parallel simulation using RTI," *In Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications,* p. 12, 2003.

[38] K. Rycerz, M. Bubak, M. Malawski, and P. M. A. Sloot, "A Framework for HLA-Based Interactive Simulations on the Grid," *SIMULATION,* vol. 81, no. 1, pp. 67 - 76, 2005.

[39] K. Rycerz, M. Bubak, M. Malawski, and P. Sloot, "Grid support for HLA-based collaborative environment for vascular reconstruction," *In Proceedings of the Second IEEE International Conference on e-Science and Grid Computing,* p. 48 , 2006.

[40] K. Rycerz, M. Bubak, P. Sloot, and V. Getov, "Problem solving environment for distributed interactive applications," *In Proceedings of the CoreGRID Integration Bibliography 246 Workshop,* pp. 129 - 138 , 2005.

[41] K. Rycerz, A. Tirado-Ramos, A. Gualandris, S. F. P. Zwart, M. Bubak, and P. M. A. Sloot, "Interactive n-body simulations on the grid: HLA versus MPI," *ternational*

Journal of High Performance Computing Applications, vol. 2, no. 21, pp. 210 - 221 , 2007.

[42] Y. Xie, Y. M. Teo, W. Cai, and S. J. Turner, "Service provisioning for HLA-based distributed simulation on the grid," *In Proceedings of the 19th IEEE/ACM/SCS Workshop on Principles of Advanced and Distributed Simulation,* pp. 282 - 291 , 2005.

[43] C. Kim, T. Lee, S. Hwang, and C. Jeong, "Grid-based parallel and distributed simulation environment," *Parallel Computing Technologies,* vol. 2763, pp. 503 - 508, 2003.

[44] T. Lee, S. Yoo, and C. Jeong, "Design and implementation of GPDS," *In Proceedings of the 4th International Conference Computational Science,* pp. 873 - 880 , 2004.

[45] R. Minson and G. Theodoropoulos, "Distributing RePast agent-based simulations with HLA," *Concurrency and Computation: Practice & Experience,* vol. 20, pp. 1225 - 1256 , 2004.

[46] G. Theodoropoulos, Y. Zhang, D. Chen, R. Minson, S. J. Turner, W. Cai, Y. Xie, and B. Logan, "Large scale distributed simulation on the grid," *In Proceedings of the Sixth IEEE International Symposium on Cluster Computing and Bibliography 247 the Grid,* p. 63, 2006.

[47] Y. Zhang, G. Theodoropoulos, R. Minson, S. J. Turner, W. Cai, and Y. Xie, "Grid-aware large scale distributed simulation of agent-based systems," *In Proceedings of the European Simulation Interoperability Workshop of Simulation Interoperability Standards Organisation and Society for Computer Simulation International,* 2005.

[48] L. Bononi, G. D'Angelo, and L. Donatiello, "HLA-based adaptive distributed simulation of wireless mobile systems," *In Proceedings of the seventeenth Parallel and Distributed Simulation ,* pp. 40 - 49 , 2003.

[49] N. Collier, "RePast: An Extensible Framework for Agent Simulation," *Natural Resources and Environmental,* vol. 8, no. 4, 2003.

[50] D. R. Jefferson, "Virtual time," *ACM Transactions on Programming Languages and Systems,* vol. 7, pp. 404 - 425 , 1985.

[51] R. E. De Grande and A. Boukerche, "Dynamic partitioning of distributed virtual simulations for reducing communication load," *In Proceedings of the IEEE International Workshop on Haptic Audio visual Environments and Games,* pp. 176 - 181, 2009.

[52] A. Boukerche and A. Fabbri, "Partitioning parallel simulation of wireless networks," *In Proceedings of the 32nd conference on Winter simulation,* pp. 1449 - 1457 , 2000.

[53] R. M. Fujimoto, "Parallel discrete event simulation," *Communications of the ACM,* vol. 33, pp. 30 - 53 , 1990.

[54] J. S. Steinman, C. A. Lee, L. F. Wilson, and D. M. Nicol, "Global virtual time and distributed synchronization," *ACM SIGSIM Simulation Digest,* vol. 25, pp. 139 - 148 , 1995.

[55] M. R. Jiang, S. P. Shieh, and C. L. Liu, "Dynamic load balancing in parallel simulation using time warp mechanism," *In Proceedings of the 1994 International Conference on Parallel and Distributed Systems,* pp. 222 - 229 , 1994.

[56] C. Burdorf and J. Marti, "Load balancing strategies for time warp on multi-user workstations," *The Computer Journal,* vol. 36, no. 2, pp. 168 - 176, 1993.

[57] R. Schlagenhaft, M. Ruhwandl, C. Sporrer, and H. Bauer, "Dynamic load balancing of a multi-cluster simulator on a network of workstations," *In Proceedings of the 9th workshop on Parallel and distributed simulation,* pp. 175 - 180 , 1995.

[58] E. Deelman and B. K. Szymanski, "Dynamic load balancing in parallel discrete event simulation for spatially explicit problems," *In Proceedings of the 12th workshop on Parallel and distributed simulation,* pp. 46 - 53 , 1998.

[59] M. Choe and C. Tropper, "On learning algorithms and balancing loads in time warp," *In Proceedings of the 13th workshop on Parallel and distributed simulation,* pp. 101 - 108 , 1999.

[60] M. Y. H. Low, "Dynamic load-balancing for BSP time warp," *In Proceedings of the 35th Annual Simulation Symposium,* pp. 267 - 274 , 2002.

[61] P. Peschlow, H. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," *In Proceedings of the 21st Workshop on Parallel and Distributed Simulation,* pp. 219 - 228 , 2007.

[62] K. M. Chandy and J. Misra, "Distributed simulation: A case study in design and verification of distributed programs," *IEEE Transactions on Software Engineering,* vol. SE, no. 5, pp. 440 - 452, 1979.

[63] J. Misra, "Distributed discrete-event simulation," *ACM Computing Surveys,* vol. 18, pp. 39 - 65 , 1986.

[64] A. Boukerche and S. K. Das, "Dynamic load balancing strategies for conservative parallel simulations," *In Proceedings of the 11th Workshop on Parallel and Distributed Simulation,* pp. 32 - 37 , 1997.

[65] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary, "Scheduling critical channels in conservative parallel discrete event simulation," *In Proceedings of the 13th workshop on Parallel and distributed simulation,* pp. 20 - 28 , 1999.

[66] B. P. Gan, Y. H. Low, S. Jain, S. J. Turner, W. Cai, W. J. Hsu, and S. Y. Huang, "Load balancing for conservative simulation on shared memory multiprocessor systems," *In Proceedings of the 14th workshop on Parallel and distributed simulation,* pp. 139 - 146 , 2000.

[67] A. Boukerche and C. Tropper, "A static partitioning and mapping algorithm for conservative parallel simulations," *In Proceedings of the 8th workshop on Parallel and distributed simulation,* pp. 164 - 172 , 1994.

[68] A. Boukerche and S. K. Das, "Reducing null messages overhead through load balancing in conservative distributed simulation systems," *Journal of Parallel and Distributed Computing,* vol. 64, no. 3, pp. 330 - 344 , 2004.

[69] A. Boukerche, "An adaptive partitioning algorithm for conservative parallel simulation," *In Proceedings of the 15th International Parallel and Distributed Processing Symposium,* pp. 133 - 138 , 2001.

[70] W. Cai, S. J. Turner, and H. Zhao, "The resource sharing system: dynamic federate mapping for hla-based distributed simulation," *In Proceedings of the 6th*

International Workshop on Distributed Simulation and Real-Time Applications, pp. 7 - 14, 2002.

[71] G. Tan and K. C. Lim, "Load distribution services in HLA," *In Proceedings of 8th IEEE Distributed Simulation and Real-time Applications,* pp. 133 - 141 , 2004.

[72] S. Fnfrocken, "Transparent migration of java-based mobile agents: capturing and reestablishing the state of java programs," *In Proceeding of the Second International Workshop on Mobile Agents,* p. 26 – 37, 1998.

[73] Y. Artsy and R. Finkel, "Designing a process migration facility: the charlotte experience," *IEEE Computer Society,* vol. 22, no. 9, pp. 47 - 56 , 1989.

[74] Z. Yuan, W. Cai, M. Y. H. Low, and S. J. Turner, "Federate migration in hla-based simulation," *In Proceedings of the 4th International Conference Computational Science,* pp. 856 - 864 , 2004.

[75] G. Tan, A. Persson, and R. Ayani, "HLA federate migration," *In Proceedings of the 38th Annual Simulation Symposium,* pp. 243 - 250 , 2005.

[76] A. Boukerche and R. E. De Grande, "Optimized federate migration for large-scale HLA-based simulations," *In Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications,* pp. 227 - 235, 2008.

[77] Z. Li, W. Cai, S. J. Turner, and K. Pan, "Federate migration in a service oriented HLA RTI," *In Proceedings of the 11th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications,* pp. 113 - 121, 2007.

[78] R. Shah, B. Veeravalli and M. Misra, "Estimation Based Load Balancing Algorithm for Data-Intensive Heterogeneous Grid Environments," *In Proceedings of the 13th International Conference High Performance Computing,* pp. 72-83, 2006.

[79] R. E. De Grande and A. Boukerche, "A dynamic, distributed, hierarchical load repartitioning for HLA-based simulations on large-scale environments," *In Proceedings of the 16th international Euro-Par conference on Parallel processing,* pp. 242 - 253, 2010.

[80] R. E. De Grande and A. Boukerche, "Dynamic balancing of communication and computation load for HLA-based simulations on large-scale distributed systems," *Journal of Parallel and Distributed Computing,* vol. 71, no. 1, pp. 40 - 52, 2011.

[81] R.E. De Grande and A. Boukerche, "Predictive dynamic load balancing for large-scale HLA-based simulations," in *IEEE/ACM 15th International Symposium in Distributed Simulation and Real Time Applications*, 2011.

[82] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni and D. Turgut, "Routing protocols in ad hoc networks: A survey," *Computer Networks,* vol. 55, no. 13, pp. 3032 - 3080, 2011.

[83] A. Boukerche, H. A. B. F. Oliveira, E. F. Nakamura and A. A. F. Loureiro, "Vehicular Ad Hoc Networks: A New Challenge for Localization-Based Systems," *Computer Communications,* vol. 31, no. 12, pp. 2838 - 2849, 2008.

[84] R. Alkharboush, R. E. De Grande and A. Boukerche, "Federate Migration Decision-Making Methods for HLA-Based Distributed Simulations," *In Proceedings of the Distributed Simulation and Real Time Applications,* pp. 190 - 197, 2014.

[85] R. Alkharboush, R. E. De Grande and A. Boukerche, "Load Prediction in HLA-Based Distributed Simulation Using Holt's Variants," *In Proceedings of the Distributed Simulation and Real Time Applications,* 2013.

[86] R. E. De Grande and A. Boukerche, "Distributed dynamic balancing of communication load for large-scale hla-based simulations," *In Proceedings of the Computers and Communications IEEE Symposium,* pp. 1109 - 1114 , 2010.

[87] A. Boukerche, S. Shirmohammadi and A. Hossain, "Moderating Simulation Lag in Haptic Virtual Environments," *In Proceedings of the 39th Annual Simulation Symposium ,* pp. 269 - 277 , 2006.