Improved Prediction-based Dynamic Load Balancing Systems for HLA-Based Distributed Simulations

by

Raed Alkharboush

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.C.S. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Due to the dependency of High-Level Architecture (HLA)-Based simulations on the resources of distributed environments, simulations can face load imbalances and can suffer from low performance in terms of execution time. HLA is a framework that simplifies the implementation of distributed simulations; it also has been built with dedicated resources in mind. As technology is nowadays shifting towards shared environments, the following two weaknesses have become apparent in HLA: managing federates and reacting towards load imbalances on shared resources. Moreover, a number of dynamic load management systems have been designed in order to provide a solution to enable a balanced simulation environment on shared resources. These solutions use specific techniques depending on simulation characteristics or load aspects to perform the balancing task. Load prediction is one of such techniques that improves load redistribution heuristics by preventing load imbalances. In this thesis, a number of enhancements for a prediction technique are presented, and their efficiency are compared. The proposed enhancements solve the observed problems with Holt's implementations on dynamic load balancing systems for HLA-Based distributed simulations and provide better forecasting. As a result, these enhancements provide better predictions for the load oscillations of the shared resources. Furthermore, a number of federate migration decision-making approaches are introduced to add more intelligence into the process of migrating federates. The approaches aim to solve a dependency problem in the prediction-based load balancing system on the prediction model, thus making similar systems adapt to any future system improvements.

Acknowledgements

First, I thank Allah for giving me the strength and patience to finish this amazing journey and an important chapter of my life.

I would like to thank my research supervisor Dr. Azzedine Boukerche who has been a source of support, encouragement, and guidance towards the completion of my thesis. I am grateful to him for accepting me as a student in PARADISE Lab, where I spent the most challenging days of my life.

My friends in PARADISE lab are like a second family to me. Dr. Robson De Grande, thank you for your help and the discussion sessions we have had. I wish to thank everyone in the lab, but the space is small. So, Thank you guys for everything.

My parents has given me nothing but support. They believed in me from the first time I opened up my eyes in this world. I am thankful for having such an amazing parents. I also want to thank my brothers and sister for their moral support and I wish them the best in their lives.

Two new and really important people in my life who have pushed me really hard to finish my work. My lovely wife, Bashaer, for helping me and providing a suitable environment for me in the apartment to work. The second person is my daughter, Joanna. I appreciate you waking me up early in the morning so I could start working early. I know milk was all what you needed.

Nothing of this would have happened without my company and sponsor, Saudi Aramco. They believed in me to pursue my masters when I wished to continue my studies. My management, I am grateful for your encouragement and going the extra mile to help me getting a sponsorship.

List of Publications

The following publications are relevant to the topic of this thesis:

1. Alkharboush, R.; De Grande, R.E.; Boukerche, A., "Load Prediction in HLA-Based Distributed Simulation Using Holt's Variants," Distributed Simulation and Real Time Applications (DS-RT), 2013 IEEE/ACM 17th International Symposium on ,

2. Alkharboush, R.; De Grande, R.E.; Boukerche, A., "Federate Migration Decision-Making Methods for HLA-Based Distributed Simulations" Distributed Simulation and Real Time Applications (DS-RT), 2014 IEEE/ACM 18th International Symposium on ,

Glossary

ANN Artificial Neural Networks

ARIMA Autoregression Integrated Moving Average

ARMA Autoregression Moving Average

BPTT Back Propagation Through Time

CAT Cluster Advance Rate

CLB Cluster Load Balancer

CPC Cluster-to-Process Communication

CPU Central Processing Unit

DR Direction Ratio Approach

EWMA Exponentially Weight Moving Average

FL Fuzzy Logic

FOM Federation Object Model

FTP File Transfer Protocol

GA Genetic Algorithm

GIS Grid Information Services

GRAM Grid Resource Allocation Manager

GVT Global Virtual Time

HLA High Level Architecture

IA Impact Approach

IPC Inter-Process Communication

LLB Local Load Balancer

LMS Load Management System

LRC Local RTI Component

LP Long-term Projection

LVT Least Virtual Time

MDS Monitoring and Discovery System

MIS Monitoring Information Service

MP Medium-term Projection

OMT Object Model Template

PAT Process Advance Rate

PCA Principle Component Analysis

PDES Parallel and Distributed Event Simulations

PPC Process-to-Process Communication

RA Restricted Approach

RNN Recurrent Neural Networks

RT Reversed Trend

RTI Run Time Infrastructure

RTRL Real Time Recurrent Learning

sMAPE Symmetric Mean Absolute Percentage Error

SOM Simulation Object Model

SP Short-term Prediction

SS Separate Systems

SSRT Separate Systems + Reversed Trend

SVM Support Vector Machine

VTP Virtual Time Progress

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Distributed simulations, such as large High-Level Architecture (HLA)-Based simulations, have been providing easy and fast implementations for different complex problems, such as evaluating localization protocols and algorithms [79] [31] [30] [32] [80], measuring the performance of network routing protocols [15] [20] [6], comparing the wireless sensor algorithms [11] [14] [33] [22] [21], analyzing the wireless networking protocols [10] [19] [18] , computing the consumed energy by the wireless protocols [15] [13] [85], exploring new network protocols [3] [25] [52], presenting new distributed data management systems [36] [27] [28], identifying issues with mobile systems and test new systems [35] [34] [26] [29], and ad hoc network modeling [12] [83] [11].

Distributed simulations depend on distributed and parallel/concurrent systems; this being so, several factors, which are listed below, can affect the performance of the distributed simulation: improper distribution of simulation entities among the resources, the existence of external load in the resources, resource heterogeneity, and dynamic changes in simulation load. A load balancing system can initially distribute simulation entities based on resource heterogeneity before starting the simulation. This method provides a static way to prevent load imbalances in the beginning of the simulation. However, such systems will not provide a solution to simulations with dynamic load changes that cause load imbalances afterwards. As a result, several dynamic load balancing systems have been designed in order to prevent load imbalances by continually monitoring load. Dynamic Load Balancing improves the performance of distributed simulations by dynamically redistributing simulation entities once a load imbalance is predicted.

HLA is a fully equipped framework that simplifies the implementation of distributed simulations. HLA introduces two concepts to distributed simulations that improve the

design of distributed simulations, reusability and interoperability of simulation entities. Reusability allows simulation entities (federates) to be reused in different distributed simulations (federations), where interoperability adds the capability for those simulation entities to communicate. In order to use the concepts of reusability and interoperability, HLA introduces *rules* that must be followed by federates and federations. Moreover, HLA defines *interface specifications* to allow the communication between federates and their respective federations and to allow *Object Model Templates* to publish objects and interactions used in the distributed simulation.

As HLA was built with dedicated resources in mind, HLA experiences weaknesses when it comes to an environment composed of shared resources. HLA, for example, lacks the ability to control distributed simulation entities on shared resources and a federate migration protocol that moves federates around without pausing the whole distributed simulation. Most importantly, HLA is not capable of performing any load balancing function.

Extensive work has been conducted in the field of designing systems to prevent load imbalances in distributed simulations. Such resource balancing systems have used different mechanisms in order to balance different elements in a distributed simulation environment, such as computation load [24], communication load [47], a combination of computation and communication load [63], migration load [48], and prediction [49], which uses an extension of Exponential Weighted Moving Average (EWMA), with double exponential smoothing for load forecasting, this is named Holt's model, which is one of the main interests in this thesis.

1.1 Thesis Statement

As HLA-Based simulations can face degradation in performance, load balancing schemes become an essential and a must to decrease the simulation time by re-distributing load evenly to better consume shared resources. There are some existing systems that solve this issue by providing a centralized and dynamic solutions that aim to provide load balancing capabilities for HLA-Based simulation in large-scale environments, one of which employs a prediction technique to properly re-allocate simulation entities to resources with low load CPU consumption. However, the performance is found to be dependent on the prediction technique. A number of different prediction mechanisms are considered. Therefore, this work proposes a set of suitable and different variants to a prediction model to introduce a better adaption for the used distributed load balancing systems.

Moreover, a number of federate migration approaches are developed to introduce dynamic and prediction model-independent decisions for federate migrations.

1.2 Motivation

Load imbalances can substantially decrease the performance of the distributed simulations, which put limitations on the execution of such simulations on certain resources. Therefore, a resource management system that is capable of controlling and organizing the distribution of load, optimistically or conservatively, is needed. The existing resource management systems, such as Grid Services [56], provide monitoring mechanisms for distributed resources and applications, so, a general purpose balancing system is required.

Load Balancing is a problem that exists in different areas as it is a challenging and important issue to solve. As load balancing is an Nondeterministic Polynomial (NP)-Hard problem, there exists no solution that provide an optimal solution that solves the redistribution issue in polynomial time. Balancing computational load in a number of non-dedicated resources can be solved with a distributed multiple knapsack [61], and an optimal solution to the load distributed is achieved.

To react and detect load imbalances during the execution time, balancing systems use heuristics to offer a redistribution of load in a reasonable time. Different systems use different metrics applied to the heuristics to analyze the possible migrations that lead to a better balanced environment. The system the work is based on uses Central Unit Process (CPU) consumption to identify load imbalances and react accordingly. However, the system uses a prediction model that is not well tuned to work on simulation-like loads. Therefore, a look into how the prediction model reacts to simulation-like loads is needed to modify the prediction model in a way to tune it better for such loads.

1.3 Objectives

As the work of this thesis is incorporated with a previous dynamic load balancing system, a need to understand how the system behaves was essential for proposing new improvements. The previous system uses a prediction technique, namely Holts model, to predict the resource load in different times in the future and triggers migration calls to transfer simulation entities from overloaded resources to underloaded resources. Therefore, the load behavior for running HLA-Based distributed simulations, which uses the previous

system to perform balancing capabilities, is analyzed and observed to find weaknesses in the implementation of such prediction models in similar environment. The analysis and comparison between the actual load and the projected loads demonstrates two weaknesses. These two weaknesses are related to how the parameters of the used prediction model can negatively affect the projection. Additionally, changes to the prediction model would not have an effect as the previous system is not adaptive to new prediction models or variants. Therefore, the objective of this work is to find efficient, *low computational needs*, and effective, *decent results*, ways to improve the prediction accuracy and the federate migration decision making that are proposed in the previous system.

1.4 Contribution

In order to better adjust the prediction system to deal with the weaknesses identified in running in distributed simulation, a number of variants are proposed. Each variant tries to solve a certain weakness on its own way in order to provide better projections for loads generated by running distributed simulations. *Cutoff* variant introduces the concept of threshold to overcome the unrealistic projections that are computed by the prediction model. *Reversed Trend* is developed to minimize the time needed by the prediction model to realize that a load oscillation is happening and it needs to react accordingly. *Separate Systems* breaks down the prediction model into sub-models where each is responsible for calculating a certain projection. *Genetic Algorithms* are used to dynamically adjust the prediction model parameters to add a dynamical adjustment feature to the prediction model in order to adopt to any load behavior. A combination of *Separate System* and *Reversed Trend* is introduced in another system that solves multiple problems at once.

For the federate migration decision making schemes, the previous scheme uses a static assignment to thresholds that are tuned for a certain type of prediction model. A number of dynamic schemes are introduces to remove the dependency on the statically thresholds. *Rules Approach* and *Direction Ratio* use a set of rules to see if there is a justification for a migration based on the load and the loads' tendency of the source and the destination, along other metrics. *Fuzzy Approach* is proposed to provide a more meaningful decision that is based on the sets the resources are assigned to. The *Impact Approach* gives more important to migrations that have a greater impact on the shared environment.

The work and results of this thesis are restricted on the used commodity hardware, which lacks Graphical Processing Units (GPU), and on a total of 40 compute nodes that are distributed under 2 clusters.

1.5 Outline

The thesis is organized as follows: Chapter 2 presents a background history in the field and related work. The background work describes HLA standards and its limitations. It also gives a brief about Grid Services as it is applied in the used system, which also describes the architecture that uses Grid Services to allow the implementation of HLA-Based simulations in large-scale environments. The related work describes the existing dynamic load redistribution solutions and their drawbacks. Chapter 3 lists the proposed variants and discusses the accuracy and complexity of each variant. Chapter 4 introduces a number of approaches to make the federate migration decisions smarter. The approaches are then compared in terms of behavior. In Chapter 6, experiment results are discussed and detailed. Finally, Chapter 7 presents the conclusion and future work.

Chapter 2

Background and Related Work

In this chapter, some background information is presented to give an understanding on the kind of challenges faced during designing a dynamic load balancing system for large-scale HLA-Based simulations, in paticular, the one that uses prediction to perform its functionality. HLA is described to give an idea about its limitations when it comes to load imbalances. As having a tool to manage distributed resources and applications is important, Grid Computing Services are described. These kinds of tools are used for monitoring, accessing remote resources in a distributed, shared environment. The related work presents a list of proposed approaches in load balancing distributed simulations to show their drawbacks and challenging issues.

2.1 High Level Architecture

HLA is a general purpose distributed simulation framework that eases the design of distributed simulations. The standard HLA was initially developed by the United States Department of Defense to have an efficient way to execute distributed, parallel military simulations. After the Institute of Electrical and Electronics Engineering (IEEE) standardized [1] HLA specifications in 2000, it became the recommended process for developing parallel simulations. HLA introduces management mechanisms and design principles to prevent inconsistencies and to allow reusability and introperability in distributed simulations. Reusability is the result of separation of the simulation into independent components that can be used in other simulations without any further modifications. Interoperability is the result of enforcing interaction standards to the simulation components that allows the components to operate with one and another without any changes.

HLA distribution simulation is called *federation* and it composed of a number of independent simulation components called *federates*. HLA standards are followed by the federation and its federates, and these standards consists of rules, object model templates, and interface specifications.

HLA rules defines the responsibility of federations and federates in the simulation. These rules ensure that the simulation follows the standard to avoid inconsistencies. They guide the simulation design and development into the HLA specification aspects. To comply with the HLA specification aspects, Federation Object Model document is required to define the simulation objects, simulation object exchange among federates as restricted by Run Time Infrastructure (RTI), the access and modifications of simulation objects and federate coordination of its local time.

Object Model Templates (OMT) provide means and policies for the communication within HLA simulations. The template lists the information to be exchanged during the life-time of a simulation. It describes the shared objects, their attributes and interactions. This description defines the information that are exchanged. This facilitates the interoperability among simulations and the reusability of the federates. OMT consists of *Federation Object Model* (FOM) which defines the data for the whole simulation, and *Simulation Object Model* (SOM) that defines the data for a single federate.

The Interface specifications defines management services for HLA and the Application Programming Interface (API) to use them. The management services organizes the simulation entities are used to restrict the simulation interaction in order to take the pressure off from the developers to not deal with the simulation design. Instead, they would concern only about simulation modeling. The APIs are used by the federates to allow an interaction and exchanging information with other federates in the federation.

Figure 2.1 shows the general architecture for HLA. RTI is composed of three components: RTI executive process (RTIExec), Federation Executive process (FedExec), and RTI library (libRTI) [1]. *RTIExec* organizes federations and FedExec. *FedExec* manages the one federation, which cover managing all federates within it. *libRTI* is a library that is available to all federates, as per the HLA specification, which adapts the mechanism employed in order to manage distributed simulations and provide them as management services. These services are used by the federates to organize the operations and the exchanged data. Therefore, the communication between federates are managed by RTI.

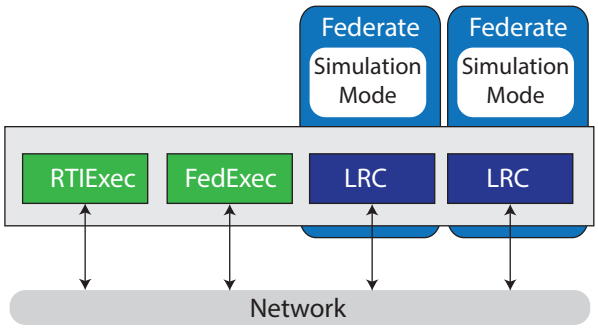


Figure 2.1: HLA Architecture

The access to these services is made possible for the federates through the use of a Local RTI Component (LRC), which contains the libRTI.

HLA has been designed considering that there is always dedicated, reliable resources with enough power for the simulation. Thus, it does not provide ways to control the distributed simulation load in shared resources. As any sudden usage for the resources for processes other than the simulation, can cause an overload, different balancing systems have been designed and proposed to provide the maximum throughput for a simulation that runs in shared resources. This is done by applying measurements to utilize the shared resources well and spread the load among them equally. The work conducted here is a set of improvements to one of the proposed systems

2.2 Grid Computing

To administer a set of distributed resources, resource management systems are needed. Grid system, through provisioning different services, is the commonly used system to manage distributed simulations that run on shared resources. Grid computing provides means to coordinate sharing resources in a more flexible and secure manner [57]. Grid computing is the use of a set of combined computer resources to execute a certain type of applications. The execution of the application is done on a set of a distributed systems on non-interactive processes and data exchange. This gives another definition for Grid Computing: A high performance computing system that runs loosely coupled, heterogeneous, and geographically scattered systems that uses message passing to interact. Grid eases the submission and management of work load to its users by providing a transparent access to its resources.

Open Grid Services Architecture (OGSA) [56] was designed by the Global Grid Forum (GGF) to equip scientific applications with a proper service-oriented grid computing architecture. OGSA provides standards that overcome the capability and behavior concerns of the casual grid computing. OGSA provides means to make use of web services in the grid architecture. OGSA ensures interoperability for different types of applications of heterogeneous systems. OGSA provides: infrastructure services, execution management services, data services, resources management services, security services, self-management services, and information services [56]. Due to its capabilities, OGSA was adopted by different major grid projects, one out of which is Globus Alliance [78]. Globus Toolkit is the most widely used middleware standard for grid computing

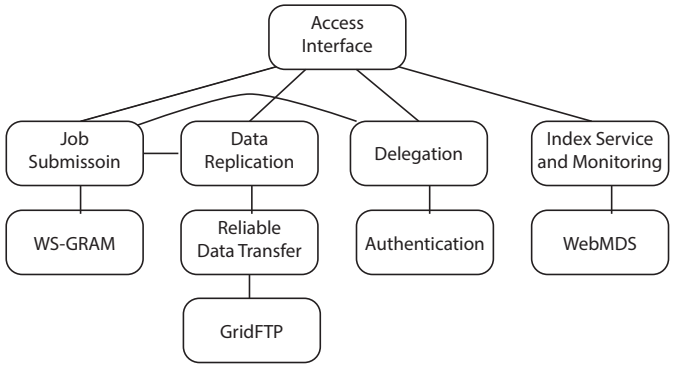


Figure 2.2: Grid Services Architecture

Figure 2.2 shows the Grid Services architecture, based on OGSA. *Access Interface* provides an easy and transparent way for users to access the grid and its services to launch their applications. *Job Submission* uses Grid Resource Allocator Manager, *WS-GRAM*, that launches jobs on particular resources, checks the execution status of the job, and returns the final result. WS-GRAM uses schedulers which identifies the best suitable resoruces to run the job. *Data Replication and Data Management* moves the information needed by the job securely to the set of resources the job was assigned to to run on. It uses *GridFTP* to securely transfer the information to the resources. *Index Service and Monitoring* provides monitoring and discovery services, *WebMDS*, that provide the status of each resource and applications. For security, system enforces *Authentication* to authorize the access to the system while maintaining data integrity.

None out of the Grid services provide means to resolve load imbalances. The reason behind mentioning Grid Computing is that it is used by different balancing systems to get the load of the resources in the grid and to perform load transfers.

2.3 Related Work

The related work is divided into two main paths: balancing systems for distributed simulations and prediction techniques. Due to the dependency of the used type of distributed load balancing system for distributed HLA-Based simulations on the prediction models, a number of prediction techniques are listed and described in this section. In order to improve the performance and efficiency of the prediction model in the balancing system, different variants are proposed by observing the implementation limitations of the prediction model.

2.3.1 Balancing Schemes for Optimistic Simulation

To solve the uneven load distribution problem for Parallel Discrete Even Simulations (PDES) that run on shared resources, a number of different dynamic load balancing approaches have been proposed. The existing approaches for distributed simulations try to recognize load imbalances and transfer load between resources to increase the performance of the distributed simulation by decreasing the simulation time. In order to achieve this goal, the existing systems evaluate the state of the simulation applications or a number of feature of the system where the application run. The existing systems use different merits in their load balancing capabilities, such as resource heterogeneity, monitoring metrics, the presence of external background load, simulation entities' interaction, simulation computing load, and other characteristics that are simulation-specific, such as lookahead and virtual time progress. None of the approaches in the literature review address what best suit HLA-Based distributed simulations that run on large-scale distributed systems.

Balancing Schemes for Optimistic Simulation

Some balancing schemes for optimistic simulations are developed based on the characteristics of optimistic parallel simulations [67] [58] [86]. Different aspects are used in this type of simulations to recognize load imbalances and re-distribute load among resources to decrease the execution time. The characteristics that are used in the balancing system can be Global Virtual Time (GVT), Virtual Time Progress (VTP), or Least Virtual Time (LVT). In this type of schemes, a number of aspects are used to recognize load imbalances and migrate simulation entities in order to decrease the overall simulation execution time.

Glazer and Tropper [62] made use of the simulation advance rate as a metric to recognize imbalances of optimistic distributed simulations. The simulation advanced rate is founded on the simulation entity's consumption of the central processing unit (CPU) and its time advance. The measurements are used to allocate a suitable CPU time slice to the simulation entity when the load is re-partitioned by the balancing system on the distributed system's resources. Their scheme goes over two phases: collecting CPU allocation and simulation advance time, and re-calculating a new proper load distribution. The load of the distributed systems are re-allocated based on the advance time, where nodes with higher advance time are nominated to get more load up to their capacity. To prevent abrupt resource re-allocation, smoothed time slices are implemented in the balancing technique. Jiang et al. mentioned [69] that the Glazer and Tropper's scheme does not offer support for load redistribution on heterogeneous systems. As a result, Jiang et al. proposed an extension that supports heterogeneous systems by the use of weights when calculating the simulation advance rate in order to consider heterogeneity aspects. These weights have to be accurate to have a proper balancing. External loads are not taken into account in both schemes, and the metrics used to monitoring the system are based on the simulation's code, which requires providing the same load always for a correct load balancing.

Burdorf and Marti [39] used the LVT of the distributed resources to build a dynamic load balancing scheme. After the initial, static load partitioning, the scheme gathers the LVT of the resources, in a peer-to-peer approach, through a vector. Based on the values in the vector, the system calculates the mean and the standard deviation of the values in the vector. With the help of the calculated values, the system defined underloaded and overloaded resources. The furthest behind simulation entity from the furthest behind resource is moved to the furthest ahead resource. In order to migrate loads, the system notifies other simulation entities about the migration, stores the message received by the migrated simulation entity during its migration, and sending the stored messages in the GVT calculations. This scheme does not deal with heterogeneity issues and shared resources. Because of its sequential data collecting mechanism, its monitoring approach does not scale. Its detection and load re-allocation are based on simple, imprecise comparisons, and its migration produces a large latency.

Schlagenhaft et. al. [82] introduced a load balancing scheme that uses static partitioning and dynamic load balancing mechanisms. The static partitioning groups the strongly connected simulation objects in simulation entities to minimize the communication overhead. The dynamic balancing part recognizes load imbalances and tries to lower

the number of simulation rollbacks by the use of the virtual time progress (VTP), which signifies the average simulation speed in a resource. The system triggers a migration when the VTP of a resource exceeds a threshold. As the migrations introduce latency to the simulation, the system measure the efficiency of a migration by computing the latency the migration introduces to the simulation. If a migration is justified, a load mover transfers the simulation object and broadcasts information about the migration to the other simulation objects. The proposed scheme does not go into details on how the migrations are generated. Moreover, the scheme does not consider heterogeneous nor shared resources.

Avril and Tropper [4] introduced a dynamic load balancing system that is aimed to solve load imbalances in clustered optimistic distributed simulations. The scheme reallocates loads evenly among the resources to achieve a better resources' utilization. The system use metrics, such as communication dependencies and a message throughput, to move load. In every iteration, the system computed the prospective processed non-rollback messages in a resource to categorize resources as overloaded and underloaded. Based on these criteria, the system pairs migrations between the different resources. The migrations between the overloaded resources that have larger inter-process communication (IPC) are chosen. The published work does not detail how the system perform chosen migrations; the scheme also does not offer a solution to the heterogeneity issues and existence of external background load.

Wilson and Shen [91] proposed an adaptive dynamic load balancing system to load changes in discrete-event simulations. In a two-level process, this technique evenly distributes simulation load among resources and minimizes the communication overhead between the simulation elements. In the first level, the system stops all processes to collect CPU consumption and pending load of a simulation. Based on the gathered information, the necessity of a load migration is decided, resources are grouped into overloaded and underloaded, and migration moves are generated based in the load processed by a processor and the pending load of the system. In the second level, load migrations are performed to achieve a load balanced environment. The system assigns weights to loads to give more importance to migrate loads with higher weights. This system does not mention how migrations are performed. Moreover, it does not go over heterogeneity issues nor the existence of external load.

Carothers and Fujimoto [41] developed a dynamic load distributor for clustered optimistic distributed simulations that run on shared resources. The proposed system measures the Processor Advance Rate (PAT) for monitoring. PAT is collected periodically,

therefore, represents the actual required time for the simulation to advance one unit. The system uses processor allocation policy and load balancing policies to determine migrations. The processor allocation policy compares the resource's CPU consumption with a set of thresholds to identify if a resource is available and can handle the simulation. The load balancing policy compares resources' PATs to find the migrations. In order to make the migrations more efficient, the system uses a threshold to minimize the number of migrations and an optimized migration technique to decrease the simulation entity's size. Such systems that use PAT and CPU consumption can detect the existence of an external background load. Heterogeneity issues are not discussed in the system. However, an extended version [42] introduces TWFrac and Theta to solve the heterogeneity issues. The system was not clear on how the load of the resources are measures. The system shows limitations when heterogeneous simulations are running.

Jiang et. al. [68] system is an extended version of Carothers and Fujimoto's [41] scheme that introduces the concept of using different aspect of communications as metrics in the load balancing system. These metrics, such as Processor-to-Processor Communication (PPC) and Cluster-to-Processor Communication (CPC), are collected by a central unit and processed to determine load imbalances. In order to perform migrations, the system integrates CPC along with CAT to nominate a simulation cluster. Through a specifically developed framework, the system uses Grid services to manage the simulation entities. As the proposed system here is an extension of [41] , the system inherits its limitations, the lack of supporting heterogeneous resources, large latency migration, and the dependency on simulation code. The proposed balancing scheme includes communication in a load balancing system previously proposed, and consequently the scheme inherits the same disadvantages, such as the lack of supporting heterogeneous resources, large latency migration, and the dependency on simulation code.

Low [73] designed a cost model that uses the communication rate, the consumption load, and the lookahead of simulations to be incorporated into a dynamic load management scheme. The management scheme calculates the load balance by comparing the cost of every simulation's superstep with a threshold. The management scheme starts by redistributing computational load, then organizing simulation entities to minimize communications, and finally the simulation entities are moves to minimize lookahead differences. The scheme does not detail the load transfer mechanism, and the metrics for communication and computation are not explained. Additionally, the scheme does not consider the heterogeneous resources nor external load.

Choe and Tropper [45] improved the performance of optimistic distribution simulations by introducing a flow control and a dynamic load balancing system. The flow control tries to reduce the differences between the simulation entities pace which the dynamic load balancing system uses, along with the memory consumption, to re-distribute loads among resources. Based on a load metric of the simulation entity, the flow control limits the optimism of the simulation by applying restrictions to the total number of messages that are allowed to be sent per simulation entity. The space-time product, as a load metric, is calculated based on the least LVT in a process and the memory consumption. The dynamic load balancing system uses the same load metric to compare the system's mean with the resource's mean to see if there is a justification to trigger a migration from an overloaded resource to an underloaded resource. The system does not go into details on how the migrations of the simulation entities are done and does not consider heterogeneous, nor shared resources in its scheme.

Deelman and Szymanski [50] proposed a dynamic load balancing scheme for parallel and distributed event simulations that solve spatial problems. This type of simulations is modeled as a ring where each simulation entities run by moving to its only two neighbours. The proposed system balances the simulation by extracting the characteristics of these simulations and use the queued number of events for processing of a simulation entity as a load metric. The system uses this number to calculate the mean to detect load imbalances by comparing the dominant load chain with the mean. The dominant load chain re-allocate some of its load from its cell to one of its neighbours' cell. To allow rollbacks, additional information is kept in previous cells. As a result of this type of load migration, this type of schemes does not deal with the issue of resource heterogeneity nor the existence of external load.

Peschlow et. al. [81] designed a dynamic load balancing mechanism that considers both computation and communication imbalances. In order to perform such balancing, metrics related to computational load and communication load are used for monitoring. The computational load metrics in the mechanism are the simulation's CPU time, simulation advancements, and the processed events. The metrics for communication load are the interaction rates of simulation entities and used to detect network overloads. After statically portion loads among resources, the system redistributes loads among resources in alternates cycles of communication and computation balancing. In computation balancing cycles, the system compares the nodes capacity and its load to recognize overloaded resources. The system identifies overloaded resources in communication cycles by analyzing the interaction rates of a simulation entity with remote entities. During

the load migration process, the system tries to transfer loads from overloaded resources to underloaded resources without overloading them. The system requires a set of thresholds and parameters to minimize the load movements. The proposed balancing scheme identifies the load changes of a resource, but as Glazer and Tropper's scheme [62], it is dependent on simulations' code.

The previous related work shows that general purpose balancing systems are not efficient with balancing schemes based on optimistic simulations. Instead of identifying overloaded resources based in the simulation execution, previous work show that more generic metrics are required to detect load imbalances.

2.3.2 Balancing Schemes for Conservative Simulations

Other load balancing systems are designed for conservative simulations [58] [43] [76]. These systems use lookahead as a metric to detect load imbalances. These balancing systems use other metrics to dynamically re-allocate loads between resources.

Boukerche and Das [16] proposed a balancing scheme that aims to improve the performance of conservative simulations by continuously spreading the load evenly among resources and reducing the number of simulation null messages. For the load metric, the system uses the CPU-length for the arriving messages. The system then compares the load metric against a threshold to identify overloaded and undedrloaded resources. The main technique in this approach is to group adjacent simulation entities in the same resource. In order to minimize the communication overhead that is caused by the balancing system, the proposed system follows a two-level design. Additionally, a three-phase migration process is used when performing load changes. The proposed approach does not touch the heterogeneity aspects or the existence of external load.

Gan et. al. [59] introduces a dynamic load balancing scheme for shared memory systems that run conservative distributed simulations. The system starts by statically partition loads among resources. This process is important for the system's efficiency as it minimizes the communication overhead by grouping simulation objects with lookahead less than a certain threshold into one group. After the initial static partitioning, the system continuously performs scheduling to re-distribute load so that simulation runs at the same speed at all times. The scheduling algorithm prioritize the simulation entities proportionally based on their simulation times, so the differences between the simulation times are decreased. As the system is developed for shared memory systems, it does not cover heterogeneity issues, the existence of background load, nor scalability issues.

Xiao et al. [93] tried to improve the event rate for conservative simulations by introducing a three-level scheduling algorithm. The system groups the related simulation entities into tasks, which are stored in different queues. A task scheduling algorithm nominates a task to run on a set of resources. Afterwards, a simulation entity scheduling algorithm, which takes place inside a task group, chooses a simulation entity based on the communication dependencies. Finally, an event scheduling is performed according to the conservative simulation algorithm. The scheme is dependent on sparse connectivity channels and substantial lookahead to allow the scheduling and to minimize the balancing overhead per each managed simulation entity. Therefore, the proposed system minimizes the communication and computational delays. Nonetheless, it does not solve the heterogeneity issues nor the shared resources.

Boukerche and Tropper [37] [9] [17] dynamic systems are designed solve the load partitioning problem in conservative simulations that run on distributed systems. The schemes use heuristic data to re-distribute simulation entities to available resources. Additionally, the system uses annealing algorithms, which are based on thermodynamic, to define the simulation's entropy to properly determine migration moves. The system detects load imbalances by comparing the resources' mean against the computational load and communication overhead for each process. The system employs thresholds to reduce the search space for partitioning and the number of iterations needed by annealing algorithm. Moreover, the thresholds are used to inform the system when a balanced state is reached. However, the proposed scheme does not detail how the monitoring and migrations are performed. In addition, the scheme does not cover the heterogeneity issues and running shared resources.

Ajaltouni et. al. [51] introduced a management system to monitor and balance peer-to-peer JXTA [1] simulations [38] based on computational loads and communication overhead. In the monitoring phase, the system collects computational and commination related data, such as the CPU consumption time by each simulation entity and the communication overhead of simulation entities to other simulation entities. In the re-allocation phase the system alternates between communication balancing cycles and computation balancing cycles. Based on set priorities, the system coordinates the alternation of cycles to be adjusted to the simulation needs. In computational-based cycles, the system compares the load of resources against the average load of the entire system. However, system follows a similar concept to what proposed by Peschlow et. al. [81] when it is a communication-based cycle. Before a migration is triggered, the system adds

[1] http://jxta.kenai.com/

the load of the simulation entity to the destination resource to check for possible load imbalances when the migration is triggered. This scheme does not consider heterogeneity issues nor the existence of external load. Moreover, the migration procedure is simple yet costly which introduces global synchronization to the simulation.

These balancing systems use more generic metrics than the ones used in optimistic simulations. With the help of these different metrics, the discussed systems were able to identify and react properly to load imbalances. However, the design of such systems limits their use in shared resources on a large-scale environment.

2.3.3 Balancing Schemes for HLA-Based Simulations

The balancing systems discussed here are integrated into the HLA framework to support load balancing capabilities to HLA-based simulations, which enabling transparent transferring of federates through migration.

Luthi and Grossmann [74] developed Resource Sharing System (RSS) to dynamically load balance HLA-based distributed simulations. The main goal of this system was to provide load balancing capabilities without interfering with the HLA simulations, modifying HLA structure, or the simulation applications. The proposed scheme only discusses the federate migration process in details while other aspects are lightly discussed. RSS monitors the load of the available resources and balances the environment based on the load of the HLA federates. In order to achieve migrations, a communication federate is used that connects to HLA interfaces to trigger migrations and to provide access to FTP server for transferring data. Because this system uses HLA interface for the federate migration, the system introduces large latency as HLA simulations are globally synchronized.

Tan and Lim [87] optimized federate migrations by developing a load distribution technique for HLA simulations. The scheme uses a Federate Wrapper to perform monitoring, re-allocating loads, and migrating federates. To advance in time, the monitoring collects the number of emitted calls from a federate to the Federate Wrapper. To improve the re-allocation process, additional data is gathered, however, no re-allocation mechanism is mentioned. To achieve load changes, a federate migration process is used which comprises of a number of queues to avoid simulation inconsistencies and minimize delays. The proposed scheme and does not take into account resource heterogeneity for load re-allocation.

Cai et. al. [40] designed a system to support executing large-scale HLA-based distributed simulations. The main component of the proposed system is a Load Management System (LMS) that forwards migration calls to simulation entities. LMS monitors the distributed resources by accessing Grid Information Service (GIS) to gather data related to the distributed resources. The proposed scheme does not discuss the used partitioning mechanism. However, the proposed scheme goes over how LMS transfers load from an overloaded resource to an underloaded resource. The scheme migrates federates by pausing the entire simulation, followed by transferring data using GridFTP, and finally using the Grid Resource Allocation Manager (GRAM) services to remotely resume the execution of the jobs. As a result, the proposed scheme uses GIS to consider external load during analysis, however it does not solve the heterogeneity issue.

Zajac et. al. [94] used the Grid Services to design a resource management system to provide a support for the execution of interactive HLA-based simulations. The main goal of the proposed system is to dynamically configure the simulation by accessing Grid Services and enabling an interface to communicate with HLA simulations' entities. Grid Services allow the system to discover simulation elements and to transmit information needed for the federate migrations. The system migrates federates by requesting the HLA to stop and start federates, which globally synchronize the simulation resulting in large delay, and use GridFTP to transmit federates' data. The proposed system covers only the federate migration process and does not mention the monitoring nor the redistribution mechanisms.

Bononi et. al. [7] developed a balancing system for parallel and distributed simulations that analyzes the communication load. Using a middleware [8] that is aimed to ease managing HLA-based simulation communication, the system recognizes imbalances by computing the ratio between the incoming and outgoing simulation messages. With the help of the computed ratio, the balancing system tries to decrease the communication between the involved resources, which would lead to reduce the overhead to the simulation execution. These types of balancing systems require a change in the implementation of HLA-simulations as HLA simulation objects are migrated instead of federates. As the proposed system analyzes the communication overhead, the system produces computation load imbalances in the environment. Moreover, the different migration paradigm requires different implementation and design for the distributed simulations.

Summary

Table 2.1 summarizes the related work and shows that all the discussed balancing systems for discrete-event simulations have some limitations that interfere with using them in large-scale distributed environments or in environments where the resources are shared.

Table 2.1: Comparison of Balancing Schemes for Discrete-Event Simulations

	Sim.	Monitoring	Re-Distribution	Migration	Hetero	External Load
Glazor & Tropper	Opt	Time Advance	Comp.	-	-	-
Jiant et al.	Opt	Time Advance	Comp.	-	Weights	-
Burdorf & Marti	Opt	LVT (vector)	Comp. Speed (StD)	Simplistic (Slow)	Partially (Indirectly)	Partially (Indirectly)
Schlagenhaft et al	Opt	VTP	Comp. pVTP + Mig.	Undefined	-	-
Avril & Tropper	Opt	Comm Throughput	Load (Comm.)	Undefined	-	-
Carothers & Fujimoto	Opt	PAT, TWFrac	Load (Policies)	Clustered	Yes	Partially (Limited)
Jiang et al.	Opt	IPC	Comp. + Comm.	Clustered (Slow)	-	-
Deelman & Szymanski	Opt	Unproc. Events	Comp. (Chains)	Neighbor	-	-
Choe & Tropper	Opt	Space-Time Prod.	Comp.	Undefined	-	-
Low	Opt	CPU Load	Comm Comp Lookahead	-	-	-
Peschlow et al	Opt	Time Advance	Comp. Comm.	-	-	-
Wilson & Shen	Opt	CPU Load	Policies (Comm/Comp)	-	-	-
Boukerche & Das	Con	CPU Load	Comm. Comp.	-	-	-
Xiao et al.	Con	Comm. Dep.	Scheduling lvl	-	-	-
Gan et al.	Con	Sim. Time	Central (Priority)	-	-	-
Boukerche	Con	Entropy	Comp. + Comm.	-	-	-
Ajaltouni et al.	Con	CPU Load	Comm. Comp.	Global Sync.	-	-
Luthi et al	HLA	-	-	Global Sync.	-	-
Zajac et al.	HLA	Grid	-	Global Sync.	-	Only Monitoring
Cai et al.	HLA	Grid	-	Global Sync.	-	Only Monitoring
Tan & Lim	HLA	-	-	Queues	-	-

2.3.4 Load Prediction Techniques

Accurate and precise load prediction techniques are important for the efficiency of load balancing systems. Load prediction helps such systems to migrate federates among resources to have a better balanced shared environment. Having a balanced shared environment can increase the performance of the simulation time by decreasing the overall execution time. Thus, it is important to look at the different prediction models [54] in the distributed simulations field and other fields in order to understand the possible models we can use. To do so, a thorough analysis is needed to determine whether the advantage the prediction model provides comply with our requirements in the distributed load balancing system, efficiency and effective.

Similar day Approach

Similar day is a method that searches historical data looking for similar characteristics to what the system is going through at the moment to predict the load. Some of these characteristics include, but not limited to, a day of the week or a date. In another word, the historical data has to be seasonal where the characteristics of the historical data repeat themselves. The prediction computed by this approach can be the a single similar reading or a linear combination of several historical readings.

Feng [55] has used the concept of similar day in his research to model a system for ionospheric forecasting. Implementing this approach is not considered challenging once historical data is available. In our experiments, historical data can be gathered as the simulation is running. However, the load of distributed simulation does not follow a seasonal pattern. The load does not follow any pattern. Thus using this approach is not suitable because of the differences of their data behavior.

Expert Systems

This is a rule-based intelligence system. The rules are heuristic in nature and are incorporated by experts in the field to a list of statements in produce prediction without any human involvement. Such systems perform best when an expert works along system developers so that the expert's knowledge is transformed properly to a prediction system. This, however, raises a set of disadvantages of such prediction methods. Firstly, the prediction system's performance can be affected if the knowledge is not clarified well to the developers. Secondly, the experts' knowledge is built based on their personal experience. Thus, each expert would provide different input to the system developers which results

in a more complex system if a system is needed to be designed based on the knowledge of a number of experts.

It has been used in a different number of fields. Ho *et al.* [65] has employed the knowledge of the operators and the system load of the Taiwan Power system to propose a knowledge-based expert system. The proposed system provided better performance than a time-series prediction model.

Due to the dependency on rules, such systems are implemented with a set of *if statements* to cover the possible rules. Embedding an expert system is not preferred in predicting the load of resources that run distributed simulations. The behavior of the load in each time the simulation runs is different. Thus, making it difficult for experts to recognize a rule that unifies the behavior of all possible scenarios in order to produce a precise projection.

Regression methods

Regression is used to find the relationships between a dependent variable and a set of independent variables. This prediction method is the most widely used in the electrical load forecasting [66] [64] [84] [44]. It is used to find a model that represents the relationship of load consumption and other factors, such as weather, season, and the type of home.

Engle *et al.* [53] presented different regression models to forecast the load of the next day. The models proposed incorporated the changes in weather, season of a holiday, and the average load.

Implementing regression models in our systems is not feasible as the only visible variable is the system's load. Thus, making building a relationship between two different elements is impossible.

Artificial Neural Network

Artificial Neural Network (ANN) has been one of the most widely used forecasting techniques in many fields, especially electric load forecasting. Neural Network have capabilities to demonstrate non-linear fitting. The output of a neural network can be linear or non-linear function of its inputs. Neural Network has a set of inputs and outputs with a number of layers in between that are composed of neurons. There are a number of other architectures that someone has to find the suitable one for their application. Other than choosing the appropriate architecture, the developer has to select the proper number

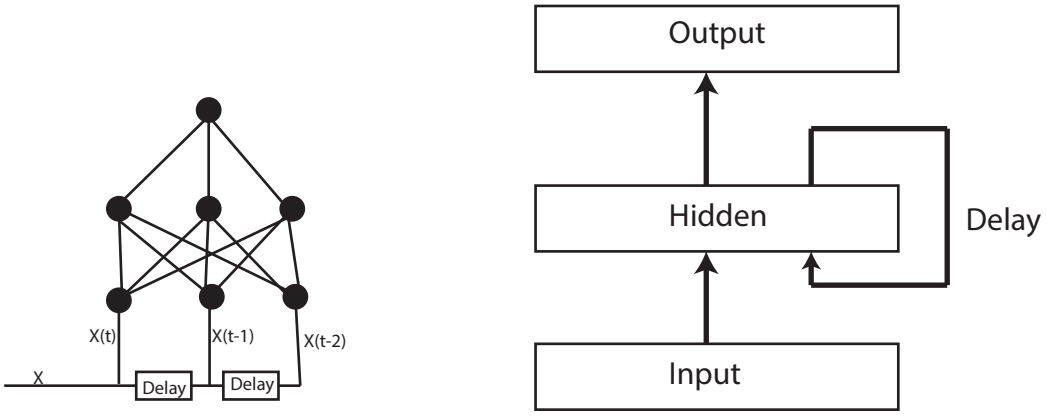
of connectivity of layers and elements, and use of uni-directional or bi-directional links. The most popular architecture in the field of electric load forecasting is back propagation [54]. Back propagation is supervised where the expected output has to be provided during the training phase so that the network can be trained properly to provide a low error output once a similar input is given. Bakirtzis *et al.* [5] implemented a neural network system for load projection for the Energy Control Center of the Greek Public Power Corporation.

To predict loads for distributed simulations, historical data is needed to better project future loads. However, casual artificial neural network do not provide meanings to use historical data. Different approaches presented to solve this issue; one of the which is tapped delay. Tapped delay is an approach where previous historical data are fed to the neural network along the current data. This approach requires an understanding of the system's requirements to know which historical data is needed to provide a decent projection. For one kind of system, the output can be related to the current load and the previous load. For another type of systems, the output can be a function of the current load and the two previous loads. Defining the set of previous data needed is challenging, especially in a field where the data does not depend of a fix-sized window of previous data.

Another solution to use historical data with neural network is Recurrent Neural Network (RNN). RNN is a type of neural network where the units are connected to form a directed cycle. This type of connection is to create an internal state that serves as an internal memory. The difference between ANN and RNN is that RNN is capable of recognizing sequences that are unrecognizable by ANN. This makes RNN suitable for pattern recognition applications. There are two main training methods for RNN: Real Time Recurrent Learning (RTRL) and Back Propagation Through Time (BPTT). RTRL does not require the system to keep any historical data. This, however, comes at a price, high time complexity. BPTT does require to keep a list of all historical data which results in reducing the overall time complexity needed for processing the output.

Support Vector Machine

Support Vector Machine (SVM) is a recent technique to solve classification and regression problems. It was inspired from Vapniks statistical learning theory [90]. SVM nonlinearly maps the data into a higher dimensional space. SVM then employs linear functions to create boundaries in the new space. Choosing a suitable kernel is an issue that is usually raised when using SVM [46]. Mohandes [77] used SVM for projection electrical load in the



(a) Tapped Delay (b) RNN

Figure 2.3: Tapped delay and Recurrent neural network Archeticture

short-term future. Mohandes compared the performance of SVM against autoregressive methods and the conclusion is that SVM was favoured over the autoregressive model. However, it needed training data to perform well.

Fuzzy Logic

Fuzzy logic is an approximation approach where inputs are assigned to sets. Based on predefined rules, fuzzy logic approximates which set the output would be in. One of the advantages of using Fuzzy Logic is that it does not need a mathematical model that maps inputs to outputs. Moreover, as inputs are assigned to sets, Fuzzy Logic does not require having a precise input for it to function. Once a solid fuzzy logic system is designed, it can give robust forecasting. As for systems that require an exact output instead of a *set*, a defuzzification process is needed to produce a precise output. This process however needs a lot of information to produce an exact output which the fuzzy logic perform operation on to get an exact output. This process is not feasible in the balancing system we are working on because of the low number of inputs we have, which produce few output sets that are not helpful in producing an exact output. Another issue is that the range of the inputs is needed to assign them to proper sets. Fuzzy has been used in a different number of electric load forecasting [70] [75].

Time-Series methods

Time series approaches assume that there is a hidden relationship between the internal structure of the data, such as autocorrelation, trend, or seasonal variation. Time series methods have the capability to understand such structure. Different methods of Time Series have been explored in different fields, such as economics, digital signal processing, and load forecasting. Among the mostly used methods are Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA). ARMA is mostly used with stationary process while ARIMA used used with non-stationary processes. For both models, they only require the time and load as inputs, and nothing else.

Liu et al. [72] compared Autoregression model against Fuzzy logic and Neural Network. Fuzzy and NN provided superb predictions for load projections in a far point in time. However, the autoregression model provided good predictions for a short period of time, in minute-by-minute predictions. Unlike Fuzzy Logic and Neural Networks, Autoregression models do not need training.

Taylor et al. [88] compared the different approaches of univariant methods (Principal Components Analysis -PCA-, double seasonal ARMA, double seasonal Holt-Winters, ANN) for short-term, half-hour and hourly, electricity load predictions. the PCA performed well, however, Holt-Winters had the best performance being a simple yet robust technique. Moreover, implementing Holt-Winters does not require a high domain knowledge. Taylor et al. [89] conducted an empirical study to evaluate the performance of different prediction models for up to a day-ahead for a seasonal data. Tayler *et al.* showed that double seasonal Holt-Winters exponential smoothing has the best performance against ARIMA and PCA for half-hour predictions.

Sarimah et al [2] conducted a study to predict the Tuberculosis cases for two years based on historical data that consists of data extracted from 72 monthly reports about the number of cases in the previous 6 years. The historical data has an increasing trend pattern without seasonality was used to determine the most suitable time-series prediction model. The authors used the number of cases for the years ranging from 2003 to 2008 to tune the listed systems and the number of cases for the years 2009 and 2010 to evaluate them. Holts model and Double Exponential Smoothing performed the best and showed a really close error rate.

Wu *et al* [92] evaluated the performance of multiple time series forecasting methods to forecast new product sales with and increasing trend. Different time series prediction

techniques were evaluated, including Exponential Smoothing model, Holt's linear model, ARMA mode, ARMA with linear trend models. Holts model performed better than Single Exponential Smoothing. However, traditional ARMA outperformed Holts model.

Summary

Based on the advantages and disadvantages of each prediction model, which is summarized in Table 2.2, Time series are the prediction model that can provide performance that match the performance we seek in our objectives. Based on the related work, Time series models provide a good short term projections, which is what we need in our system. The issue to find the most suitable time series model is discussed in some of the related work and the Holt's model shines the most as a prediction model for loads similar to simulation load.

Table 2.2: Prediction Method Summary

Method	Advantages	Disadvantages
Expert System	Easy Implementation	Subjective. Needs an expert
Similar Day	Easy Implementation	Seasonal Data. Not that accurate
Regression	Builds relationship between different elements	require an understanding of how elements are related
Time-Series	low computational needs. Understands the relationship of the internal structure	A lot of methods to choose from. Linear
Neural Network	Nonlinear. Low Error	High computational needs. Time to stabilize. No History. Choosing a suitable architecture
Support Vector Machine	Nonlinear. Solves linear and regression problems. Low Error.	Choosing a suitable kernel
Fuzzy Logic	No mathematical model. No need for a precise input. Low Error	Defuzzification

2.4 Prediction-based Dynamic load balancing system

The modifications and enhancements of Holt's prediction technique are incorporated into the prediction-based dynamic load balancing system described by De Grande and

Boukerche [49] which is presented in Figure 2.4. The load balancing process consists of monitoring the resources in order to define a set of under-loaded and over-loaded resources. Federates are then migrated to suitable shared resources in order to assure a balanced distributed simulation environment.

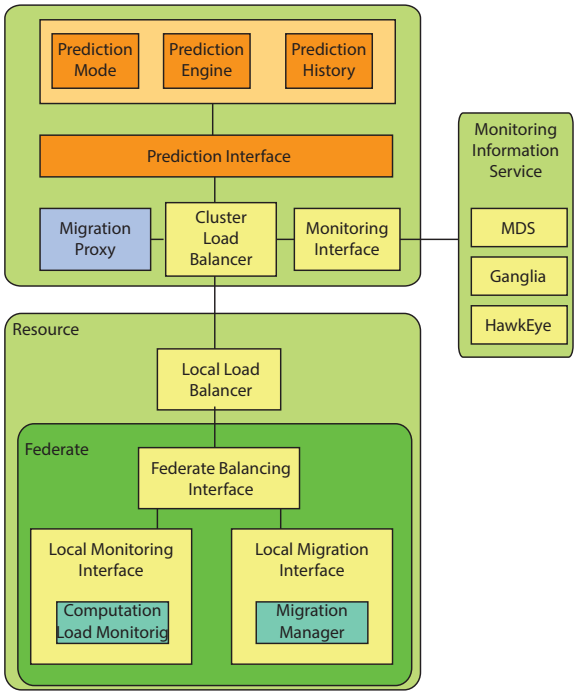


Figure 2.4: Predictive Balancing Scheme's Architecture

Each set of shared resources, within a network region and present similar capabilities, are grouped into a cluster. The cluster's master node runs a Cluster Load Balancer (CLB); that handles all shared resources in the cluster along with the federates that run on top of them. CLBs are connected with neighboring CLBs in order to exchange information concerning their own cluster. They communicate with resources through Local Load Balancers (LLB) that exist on each shared resource. Moreover, CLBs are connected to a Monitoring Interface and a Prediction Interface. The task of CLBs consists in collecting information about their shared resources and the federates they run. CLBs use the collected information in order to analyze the load and to decide on a set of migration moves. CLBs then initiate the migrations.

The Monitoring Interface provides means for CLBs to access monitored and related data of resources that are offered by Monitoring Information Service (MIS). The service can be Grids MDS, Ganglia, HawkEye, and other similar interfaces. MIS provides each

CLB with a set of load related information of concerning shared resources under its control. In order to provide such information, MIS uses Grid services. Grid is a system used to manage resources that run distributed applications over shared resources [57]. Access to load information of resources is done through a monitoring and discovery service provided by Grid Index Service.

LLBs are placed on shared resources in order to act as an interface for the resource and federates running on them. LLBs gather the load information of the simulation entities and responds to migration calls. Gathering information is triggered by a CLB, which is forwarded to a Federate Balancing Interface. The Federate Balancing Interface notifies the Local Monitoring Interface of each simulation entity, to retrieve a CPU consumption per federate. A LLB forwards migration calls, which are initiated by CLBs, to the Migration Manager.

Federate Migration is done in two steps, similarly to [23] and [71]. The migration starts with the transferring of static data and initialization files through the Grid Services to the remote resource. Secondly, dynamic execution status data and queued incoming messages are transferred from the local resource to the remote resource by the Migration Manager. Migrations performed between clusters are conducted through the Migration Proxy.

The Prediction Engine, with the assistance of Prediction History and Prediction Model, processes the incoming data. The Prediction History provides a data history which depends on the Prediction Model as each prediction model varies in data history requirements. In this model, a *smoothed value* and a *trend* are saved in the Prediction History as this work emphasizes Holt's double smoothing technique. Prediction Model provides projections based on data provided by the CLB through the Prediction Engine and data from the Prediction History.

2.4.1 Re-distribution Algorithm

The redistribution algorithm is used to re-arrange the federates in the distributed simulation to have a balanced environment, Algorithm 1 shows the flow of the system. In order to achieve this goal, the redistribution algorithm of this scheme goes through three different stages: monitoring, re-arrangement, and migration.

The distributed dynamic load balancing system periodically *monitors* the shared resources. This allows the system to be responsive to load changes. The Monitoring phase is performed periodically every Δt, which is restricted by the periodic refresh rate

offered by the monitoring tool. Δt introduces a trade-off between responsiveness and the mount of balancing overhead introduced in the system, and between responsiveness and detection load oscillations in the simulation. Having a small time interval may cause overhead to the system as a result of continuously collecting monitoring related data and perform analysis on them. In this system, Δt is set to 20 seconds, which produces a minimum overhead and awareness of load oscillations.

Algorithm 1 Distributed Dynamic Load Balancing Algorithm

```
while TRUE do
    loads ← query_MDS()
    current_loads ← filter_MDS_data(loads)
    current_loads ← normalize_loads(current_loads, benchmark)
    current_loads ← prediction(current_loads, old_loads, mig_RSCs)
    old_loads ← current_loads
    overload_cand ← select_overload(current_loads)
    spec_loads ← request_LLBs(overload_cand)
    mng_loads ← filter(current_loads, spec_loads)
    mig_moves ← local_bal(mng_loads, SP)
    mig_moves ← local_bal(mng_loads, MP)
    mig_moves ← local_bal(mng_loads, LP)
    send_migration_moves(mig_moves)
    if mig_moves = 0 then
        data_neighbours ← reques_Neighbour_Load_Data()
    else
        if relFactor ≥ random_number(1, 100) then
            data_neighbours ← reques_Neighbour_Load_Data()
        else
            data_neihgbours ← 0
        end if
    end if
    neighbours ← identify_neighbour_Less_Load()
    while neighbours ≠ 0 do
        overloaded_resource ← select(firstNeighbour, SP)
        overloaded_resource ← select(firstNeighbour, MP)
        overloaded_resource ← select(firstNeighbour, LP)
        federates ← select(spec_loads, overloaded)
        eliminate_first(neighbours)
    end while
    send_to_neightbour(overloaded_resources, federates)
    migration_moves ← wait_for_migration_moves()
    send_migration_moves(migration_moves)
    wait(Δ)
end while
```

As the projection accuracy depends on the quality of the collected data extracted from the monitoring phase, filtering is performed on the collected data to eliminate values that would affect the prediction. Moreover, the collected data is normalized to enable a fair comparison between the resource loads. Afterwards, the collected data, previous history,

and migration status are used to provide load projection in the following three different balancing cycle ranges: short-term, medium-term, and long-term. These projections are used to identify the over-loaded and under-loaded resources in any given moment.

With a knowledge of the load of shared resources and the number of projections per balancing cycle, the load balancing system applies a pair matching mechanism, shown in Algorithm 2, between the local over-loaded and under-loaded resources in each balancing cycle. The load balancing system assigns a greater importance to projections that are closer to the current time. Before the system applies the pair matching mechanism, the system first sorts the list of resources from the most over-loaded to the most under-loaded. The pair matching mechanism is performed on the ordered list.

Algorithm 2 Local Prediction Pair-Match Evaluation Algorithm

```
Require: srcRsc, dstRsc, min, θ, δ
  min, δ, θ ← adjustParameters(srcDirection, dstDirection, type)
  if dstLoad < min then
     if numberFederates(srcRsc) > 1 then
        create_migration_move(srcRsc, dstRsc)
     else
        if numberFederates(srcRsc) ≥ 1&srcLoad > (min × θ) then
           create_migration_move(srcRsc, dstRsc)
        end if
     end if
  else
     if (srcLoad − dstLoad) > (min × δ) then
        if numberFederate(srcRsc) > 1 then
           create_migration_move(srcRsc, dstRsc)
        else
           if numberFederate(srcRsc) > 1&srcLoad > (min × θ) then
              create_migration_move(srcRsc, dstRsc)
           end if
        end if
     end if
  end if
```

The mechanism initiates a migration call if the difference between the load of the over-loaded and the under-loaded resources is beyond a threshold. Based on the local migrations and their success rate, an inter-domain migration process is started. Inter-domain migration is initiated by a CLB requesting a Cluster Load from its neighboring CLBs. The Cluster Load of a CLB is the average of the loads of its resources. Once the Cluster Load is received by the initiator, the initiator will perform a selection mechanism to identify load imbalances between resources. Resources that present loads that exceed a threshold are considered as candidates for the remote redistribution mechanism functioning between domains. Once a neighboring CLB receives the list of candidates, it

performs an inter-domain redistribution mechanism, which is shown in Algorithm 3, on the three balancing cycles by comparing remote over-loaded resources and local under-loaded resources. Similar to the process of local balancing, a migration call is generated once the difference in load between the two resources justifies a load imbalance by exceeding a threshold. At the end of the selection process, all migration calls are sent at once to the requester CLB.

Algorithm 3 Local Prediction Pair-Match Evaluation Algorithm

Require: $intRsc, extRsc, min, \delta, type$
  $min, \delta \leftarrow adjustParameters(srcDirection, dstDirection, type)$
  if $intLoad < min$ **then**
    $create\_migration\_move(intRsc, extRsc)$
  else
    if $(srcLoad - dstLoad) > (min \times \delta)$ **then**
      $create\_migration\_move(srcRsc, dstRsc)$
    end if
  end if

2.4.2 Forecasting Load Status

The distributed load balancing system performs on three different load forecasting levels: short, medium, and long term. The three projections are independent from the prediction model. Each projection is designed to provide a goal to the system. Short term projection (SP) aims to solve current load imbalances while medium and long term projections (MP and LP) are used to prevent load imbalances that might happen. Each projection represents the prediction of load in certain balancing cycles. Short, medium, and long term cycles are then defined as 1, 3, and 5 balancing cycles, respectively.

The threshold used to compare the overloaded and underloaded resources is adaptive, and it is modified based on the direction of the tendency of a resource load. The amount of adjustment needed is proportional to the balancing cycle. Thus, medium and long term projections receive larger adjustments.

2.4.3 Prediction Models

The computational load of each shared resource is collected in a list recorded in fixed time intervals. This list represents the load behavior of each resource in time. Thus, time series prediction methods are used in order to forecast loads of the shared resources.

Exponentially Weighted Moving Average (EWMA) is a well-known time series prediction technique that has been used in different studies. This technique observes the

internal relationship of elements in three different aspects [60]. Single smoothed EWMA is an exponentially averaging technique that computes the smoothed value of the predicted term. Double EWMA adds a trend to the exponentially calculated average. The trend of predicting the load of shared resources is used to identify the tendency of the resource load. Triple EWMA requires a deep understanding of the system in order to detect the seasonality in the time-series data, along with the trend, and to use this knowledge to properly predict the load. Throughout the process of tuning and evaluating the performance of the proposed variants, the distributed simulation showed no sign of seasonality in its time series. Thus, the Triple smoothed EWMA is not a suitable method for forecasting load of HLA-Based distributed simulations.

Holt's model, understood as a time series prediction technique represented by the Formulas 1 and 2, is applied to the collected data in order to forecast the load. This is done by computing the value of F_{m+i} in Formula 3. The double exponentially smoothing computes the predicted load on the collected list of loads. It first finds the current smoothed value, sum_i, based on the current actual load: $elem_i$. In addition, it uses the previous smoothed value, sum_{i-1}, and the previous trend, t_{i-1}, in order to add the smoothness factor. The trend enables the extrapolation of the average of the smoothed value; and, its calculation is based on the tendency of the smoothed value, $sum_i - sum_{i-1}$, and on the previous trend: t_{i-1}. The tendency of the load is defined by the sign of the trend, where a positive sign presents an increasing tendency and a negative sign shows a decreasing tendency. Formula 3 is used to foreca

$$sum_i = \alpha \times elem_i + (1 - \alpha) \times (sum_{i-1} + t_{i-1}), 0 \leq \alpha \leq 1 \tag{2.1}$$

$$t_i = \beta \times (sum_i - sum_{i-1}) + (1 - \beta) \times t_{i-1}, 0 \leq \beta \leq 1 \tag{2.2}$$

$$F_{i+m} = sum_i + m \times t_i, m \in \{1, 3, 5\} \tag{2.3}$$

$$SP = F_{i+1}, MP = F_{i+3}, LP = F_{i+5} \tag{2.4}$$

In terms of initializing Holt's model, sum_0 requires a knowledge of the sum_{-1} which is not set in the beginning of the simulation. Thus, as a precautionary step, sum_0 is set to the $elem_0$ as there exists no previous data that sum_0 can make reference to. As in the case for the trend, t_i requires a knowledge of the previous trend. When the system is initialized, t_0 is set to 0, because it is not known whether the system is increasing or decreasing. Thus, t_1 needs to be computed as $t_1 = elem_1 - elem_0$ to represent the actual trend that Holt's model can use for the following predictions.

Chapter 3

Holt's Model Variants

Holt's model is one of the most popular Time-Series prediction models that provide resealable predictions with low computational needs. It was essential to understand the possibilities at this point to improve and enhance the prediction accuracy for such systems that already use Holt's model in their architecture for forecasting. As a result, two different paths can be considered:

1. Comparing Holt's model with other types of Time-Series technique. As tempting as it seems, different studies show that Holt's model has outperformed a number of time-series prediction techniques for similar problems. Therefore, following this path in the thesis seems a deadend.

2. Looking into different improvements to Holt's model to make the model better adapt to the behavior of distributed simulation loads. It was interesting to see how the different parameters affect changes to the Holt's model variables, especially under oscillated load.

Therefore, this thesis follows the second path to find ways to improve the prediction accuracy of Holt's model when dealing with distributed simulation load.

3.1 Holt's model's weaknesses with distributed simulation load

To define the improvements needed for Holt's model, a deep analysis was needed to define the effect of oscillated load to the internal computed variables of Holt's mode, the

computed smoothed value and the trend. In order to find areas to improve, data samples were collected from the cluster by running real simulations. Holt's model's variables were evaluated to find how they react towards different load scenarios. After looking into the results, two different weaknesses were identified with how Holt's model deals with loads similar to the load generated by running distributed simulation.

The first issue is based on how the load projection of a load in a future balancing cycle has a linear relationship with the time interval of the future balancing cycle that we need to have a load projection for. By looking at the forecasting formula of Holt's model:

$$F_{i+m} = sum_i + m \times t_i, m \in \{1, 3, 5\}$$

The time interval, m, has a linear relationship with the computed trend, which is added to the computed smoothed value. For , i, a point in time, and a fixed sum_i and t_i, the computed projections (SP, MP, and LP) will follow a linear line, as Figure 3.1 demonstrates the linear line in black color. Figure 3.1 shows a sample of the load of a node and the three different projections computed by Holt's model at each balancing cycle.

Following this linear line to calculate far-in-the-future projection is reasonable for load with not-to-often oscillations. However, as oscillations is one of the characterstics of distributed load simulation, this raises a limitation in its usage for such environment. The furthest the projection is from the current cycle, the more unrealistic the projection becomes. To provide better projections, two different approaches are proposed, Cutoff and Separate Systems.

The second issue is related to the slow response of Holt's model to adapt itself to sudden load oscillations. After looking into the Holt's model and tracking its different variables, it was noticed that the computed trend, t_i, does not always represent, or match, the actual tendency.

$$t_i = \beta \times (sum_i - sum_{i-1}) + (1 - \beta) \times t_{i-1}, 0 \leq \beta \leq 1$$

The sign of t_i represents the direction of the computed smoothed load. A t_i with a positive sign means that the tendency of the computed smoothed value is increasing. A negative value means that smoothed values is decreasing. The value of the computed trend shows how steep the tendency is.

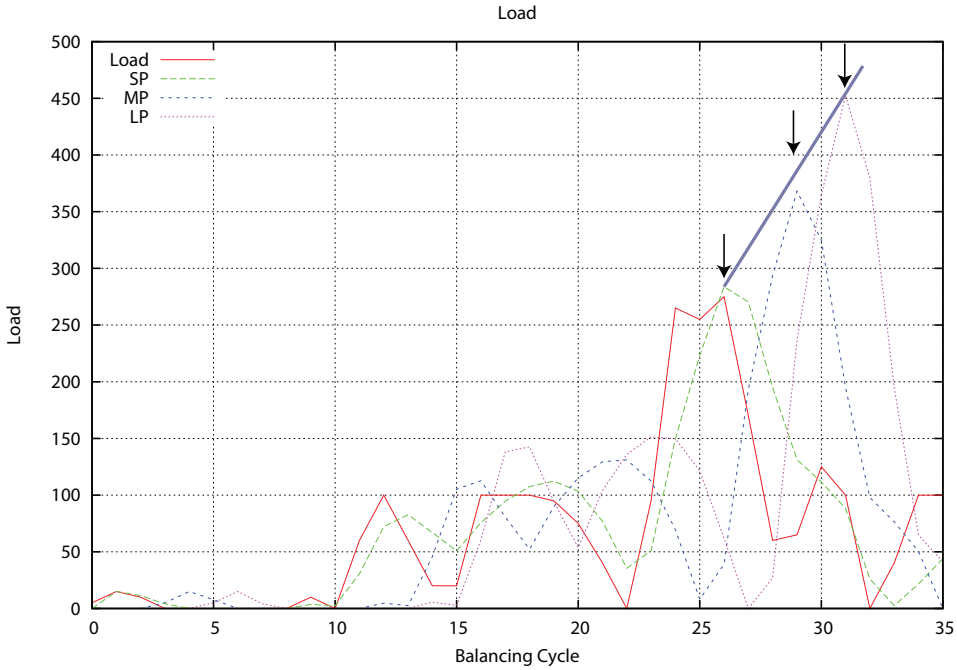


Figure 3.1: Representation of the first issue of Holt's model

By tracking the changes of t_i, t_i was slowly changing signs and tendency. The reason for the slow response of the dependency the previous computed tendency, t_i. From the previous formula, someone can conclude that changing β can help in having a more reactive system that better detects load oscillations and react properly. However, changing the value of β in our case is critical as it is used to computed furthest projections. In the comparison section, we go into details to find the best combination of α and β.

Another approach is to manipulate with the computed trend in a way to have a more reactive system. Two approaches are proposed to solve this matter, Reversed Trend and Genetic Algorithm.

Figure 3.2 shows the short-term projection which is computed by Holt's model on a data sample. At balancing cycle 13, the load oscillates and starts increasing. However, Holt's model starts adapting and gives a projection that increases after a couple of balancing cycles.

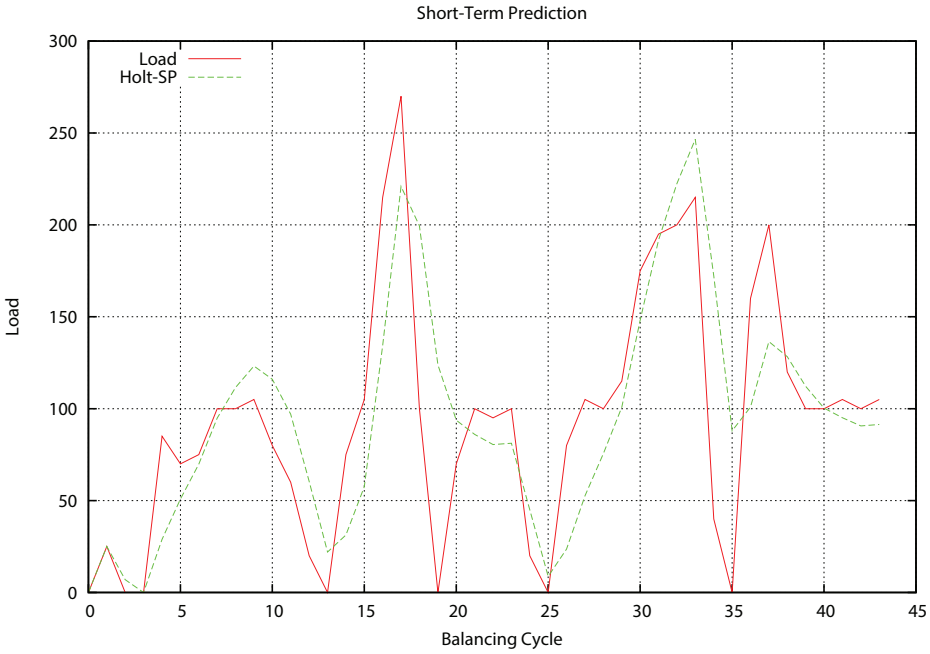


Figure 3.2: Representation of the second issue of Holt's model

3.2 Proposed Solutions

After deeply observing the issues and weaknesses in implementing Holt's model as a prediction method for distributed load balancing system for HLA-Based systems, a number of proposals are explained in detail. Each proposal provides a way to overcome the weakness it tries to solve differently. A test is conducted to compare their accuracy when dealing with distributed simulation data.

3.2.1 Cutoff

Cutoff is a naive approach that solves the first issue. When there exist load oscillations and a computed trend that has a tendency in the same direction of the load oscillations, Holt's model produces unrealistic projection values. These values depend on the steepness of the linear line that has a linear relationship between the time interval of the projection, m and the value of the projection, F_{i+m}, as shown in Equation 2.3. If the linear line is quite steep, further projections would have unrealistic and unreliable values. These values are considered for us unrealistic because of their high values that a normal system would not reach. Thus, there was a need to prevent Holt's model to

give unrealistic and unreliable projections. Cutoff tries to put some restrictions on the projections in a way to prevent these high projections. This is done using a *threshold*.

At the initial state of the system, the threshold is set to the current load value. At each balancing cycle, the system would use Holt's model to compute the three projections (SP, MP, and LP). Once they are computed, the system would compare the projection value against the threshold. If the projection has a value that is higher than the threshold, the projection's value is changed to the threshold value. This assures the system that the projections are not going to produce unrealistic values for the projections. The value of the threshold is not considered a high, or unrealistic, as it represents the highest value the system has come across. The following three equations list additional restrictions to Holt's model in a way to employ the threshold in its implementation.

$$sum_i = \alpha \times elem_i + (1 - \alpha) \times (sum_{i-1} + t_{i-1}), 0 \leq \alpha \leq 1 \tag{3.1}$$

$$t_i = \beta \times (sum_i - sum_{i-1}) + (1 - \beta) \times t_{i-1}, 0 \leq \beta \leq 1 \tag{3.2}$$

$$F_{i+m} = sum_i + m \times t_i, m \in \{1, 3, 5\} \tag{3.3}$$

$$SP = min(F_{m+1}, threshold) \tag{3.4}$$

$$MP = min(F_{m+3}, threshold) \tag{3.5}$$

$$LP = min(F_{m+5}, threshold) \tag{3.6}$$

An implementation of the Cutoff on a data sample is presented in Figure 3.3. The actual load of the node is presented in a green line, the LP projection computed by Holt's model is in red, and the LP projection computed by Cutoff is in blue. The implementation shows that when the load of the resource suddenly oscillates, the projection is *cut off* to reduce the total error that represents the differences between the actual load and the projection value.

With the implementation of the threshold, the Cutoff is expected to decrease the average error rate generated for medium and long-term predictions; this is the case because this technique attempts to prevent unrealistic projections, especially when the load oscillates, by limiting projections using the threshold. Cutoff, however, might limit the ability of Holt's model to predict short-term projections as it will create a hidden relationship between the current projection and the threshold. This behavior might decrease the accuracy of the projections for short-term projections.

The degradation experienced in calculating the short-term projections would increase performance problems when the variant is implemented on the dynamic load balancing

system. As the current dynamic load balancing system assigns more importance to short-term predictions, when generating the migration list, the incorrect computations for the short-term projections would result in an imbalanced distributed environment.

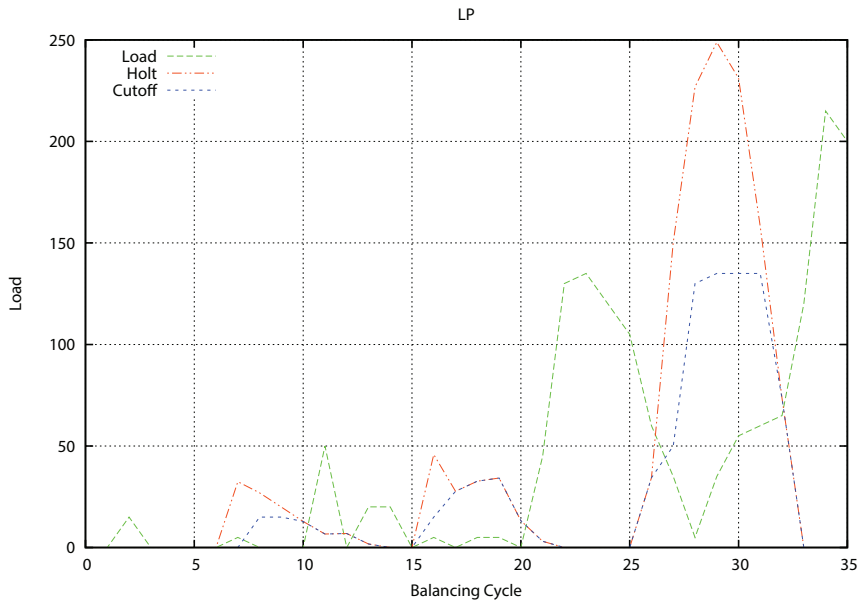


Figure 3.3: Cutoff and Holt's computation of LP

3.2.2 Reversed Trend

Reversed-Trend is a proposed approach aiming to make the dynamic load balancing system respond to load oscillations faster. A deep analysis of the behavior of the trend shows that the trend responds slowly to load oscillations. This is a dependency problem, as the t_i depends on t_{i-1}, as shown in Equation 2.2. By looking at Figure 3.2, it can be noted that the predicted load computed by Holt's model, the one in green, does not exactly follow the real load, red line, when there are sudden load oscillations.

Upon looking at the trend calculated by Holt's model for the prediction of the load, it was observed that the sign of the the trend does not change immediately in a way that reflects the changes in the actual tendency. The sign of the trend is important as it represents the tendency of the load. Moreover, it is used to predict the load in future time intervals. Thus, reverse-trend changes the sign, tendency, of the trend to match the actual tendency.

$$sum_i = \begin{cases} \alpha \times elem_i + (1-\alpha) \times (sum_{i-1} - t_{i-1}), 0 \leq \alpha \leq 1 & \text{if computed tendency} \neq \text{real tendency} \\ \alpha \times elem_i + (1-\alpha) \times (sum_{i-1} + t_{i-1}), 0 \leq \alpha \leq 1 & \text{if computed tendency} = \text{real tendency} \end{cases} \tag{3.7}$$

$$t_i = \beta \times (sum_i - sum_{i-1}) + (1-\beta) \times t_{i-1}, 0 \leq \beta \leq 1 \tag{3.8}$$

$$F_{i+m} = sum_i + m \times t_i, m \in \{1, 3, 5\} \tag{3.9}$$

$$SP = F_{m+1} \tag{3.10}$$

$$MP = F_{m+3} \tag{3.11}$$

$$LP = F_{m+5} \tag{3.12}$$

This approach compares the computed tendency of Holt's model, t_{i-1}, at the current load with the real tendency $elem_i - elem_{i-1}$. If the directions of the tendency do not match, the reverse-trend technique will *reverse* the tendency of the computed trend, that is used to compute the smoothed value; this will be done to match the real tendency. Before the inversion, the Reverse Trend first calculates the differences in load. The reason for this calculation is to prevent *unnecessary sign reversion* in small load oscillations. The reverse trend is not applied for small load oscillations as they might represent slow temporary load changes and might return to normal in a couple of balancing cycles.

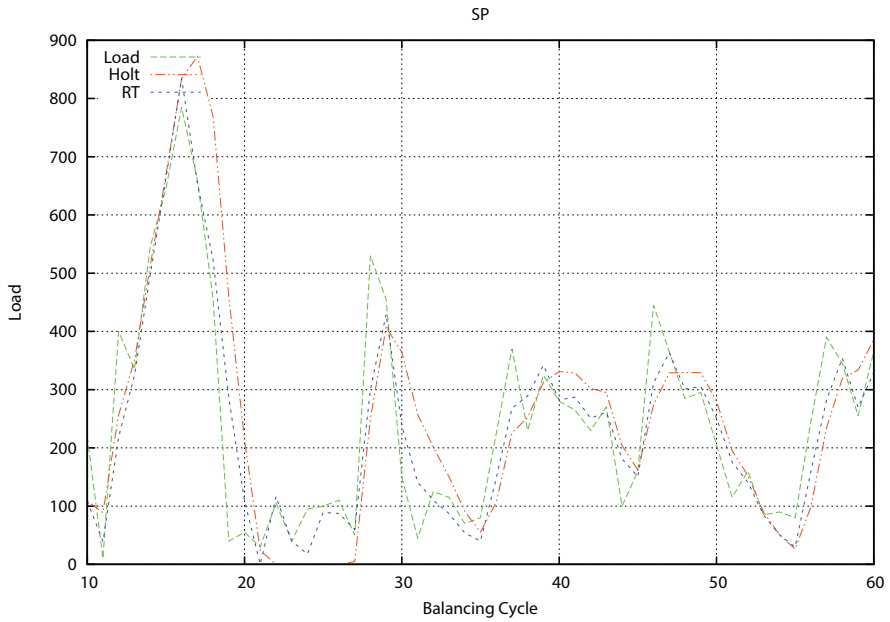


Figure 3.4: Computed SP by Holt's model and RT

Figure 3.4 shows the comparison between implementing Holt's model against RT and computing SP for a data sample. Reversing the trend sign makes the RT responds to load oscillations faster than the unmodified Holt's model.

As Reverse Trend adopts more sensitivity while reacting towards load oscillations and sharpness in projections, it is expected to provide better readings for short and medium-term projections. However, the performance achieved while projecting long-term loads might decrease; this is because the long-term projections are needed to be as smooth as possible without being sensitive to load oscillations.

3.2.3 Separate Systems

Separate Systems is an approach that aims to solve the first problem described in Holt's model. Holt's model uses a fixed α and β in order to calculate the prediction of each time interval. α represents the percentage of effect the previous smoothing values have on the current smoothing value. From Equation 2.1, the larger α is, the greater the impact is of $elem_i$ on the smoothed value and the smaller impact of the previous smoothing value is on the current smoothed value.

From observations, short term projection depends mostly on $elem_i$. On the other hand, medium and long term predictions depend more on previous smoothing values: sum_{i-1}. Thus, different fixed constants for each projection of SP, MP, and LP are used to apply emphasis on each type of prediction. This results in having three different systems, where each system is responsible for calculating a projection. The short term prediction system would have a larger value of α to exercise more influence of $elem_i$ to sum_i. Unlike the short term prediction system, the medium term prediction system presents a smaller α to emphasis sum_{i-1}. The long term prediction system then is set to present a smaller α than the medium term prediction system.

$$sum_i^{SP} = \alpha^{SP} \times elem_i + (1 - \alpha^{SP}) \times (sum_{i-1}^{SP} + t_{i-1}^{SP}), 0 \leq \alpha^{SP} \leq 1 \tag{3.13}$$

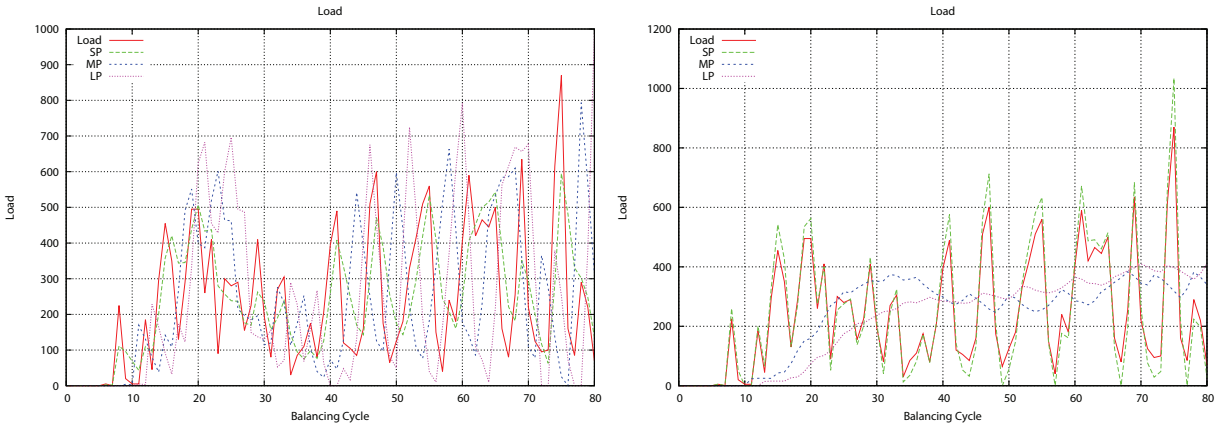
$$t_i^{SP} = \beta^{SP} \times (sum_i^{SP} - sum_{i-1}^{SP}) + (1 - \beta^{SP}) \times t_{i-1}^{SP}, 0 \leq \beta^{SP} \leq 1 \tag{3.14}$$

$$SP = sum_i^{SP} + t_i^{SP} \tag{3.15}$$

$$sum_i^{MP} = \alpha^{MP} \times elem_i + (1 - \alpha^{MP}) \times (sum_{i-1}^{MP} + t_{i-1}^{MP}), 0 \leq \alpha^{SP} \leq 1 \tag{3.16}$$

$$t_i^{MP} = \beta^{MP} \times (sum_i^{MP} - sum_{i-1}^{MP}) + (1 - \beta^{MP}) \times t_{i-1}^{MP}, 0 \leq \beta^{MP} \leq 1 \tag{3.17}$$

$$MP = sum_i^{MP} + 3 \times t_i^{MP} \tag{3.18}$$



(a) Holt's model (b) SS

Figure 3.5: The three different projections computed by each Holt's model and SS

$$sum_i^{LP} = \alpha^{LP} \times elem_i + (1 - \alpha^{LP}) \times (sum_{i-1}^{LP} + t_{i-1}^{LP}), 0 \leq \alpha^{LP} \leq 1 \tag{3.19}$$

$$t_i^{LP} = \beta^{LP} \times (sum_i^{LP} - sum_{i-1}^{LP}) + (1 - \beta^{LP}) \times t_{i-1}^{LP}, 0 \leq \beta^{LP} \leq 1 \tag{3.20}$$

$$LP = sum_i^{LP} + 5 \times t_i^{LP} \tag{3.21}$$

The Separate Systems is expected to provide better performance for all projections. This is a result of providing a different set of α and β parameters for each system that reflects the importance of the previously smoothed value and trends for each range of forecasting.

3.2.4 Genetic Algorithm

This variant uses Genetic Algorithms to solve the second issue. The basic principle is that it adopt the usage of Genetic Algorithms at each balancing cycle in order to dynamically adjust Holt's parameters, α and β. Changes to the parameters affect the performance of Holt's model and how it react to changes to the load, more specifically load imbalances. This is what we are aiming for.

Genetic Algorithm runs in iterations where certain type of processing is done on chromosomes to find a chromosome that has characteristics that fits our model the best. Each chromosome in our example is 12-bit in length that is divided in half. Figure 3.6 shows a presentation of the chromosomes. Each half is a binary representation Holt's model's parameters, α and β. The binary representation is parsed to integers and then divided by 10^6 to get a 6-digit precision value that are between 0 and 1.

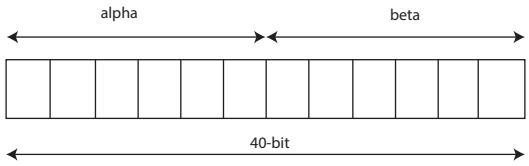


Figure 3.6: Representation of Genetic Algorithm Chromosome

Algorithm 4 shows the flow this variant follows to work. First, the smoothed value and the trend is computed using values of α and β. Then stores different values in an array that keeps track of 3 different values (current load, previous smoothing, previous trend) for the three different projection (SP, MP, and LP) for the last 6 balancing cycles in a 6-by-9 array, as shown in 3.7. This array is important as it directs this variant to properly change the values of Holt's parameters. Afterwards, the GA kicks in.

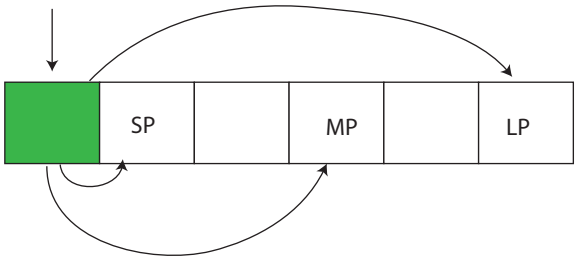


Figure 3.7: Constructing GA Array

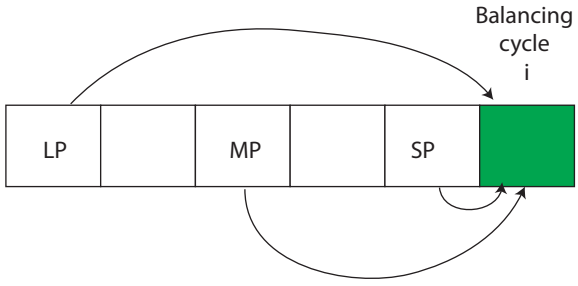


Figure 3.8: Using GA Array

GA runs over the set of chromosomes to calculate the fitness. The fitness is calculated with the help of the 6-by-9 array, where for a balancing cycle i, as shown in 3.8, the long term projection is calculated by balancing cycle $i - 5$, the medium term projection is

calculated by $i-3$, and the short term projection is computed by balancing cycle $i-1$. To calculate the fitness, all chromosomes are parsed to give numerical values for α and β. Afterwards, the algorithm makes use of the previous smoothed value and trend per previous balancing cycle to calculate a projection. These projections of these previous data should match the current load of the node. Therefore, the fitness is recognized as the difference between the projections and the current load. GA will go through all chromosomes to extract α and β to compute the projections, then the difference is computed against the current load, as shown in following equation:

$$f = \frac{3 \times (\mid current - SP \mid) + 2 \times (\mid current - MP \mid) + (\mid current - LP \mid)}{6} \tag{3.22}$$

Because of the importance to the SP projections, the difference between SP and MP is given the highest weight, followed by MP, and the lowest weight for LP.

Algorithm 4 Genetic Algorithm Approach

Require: $\alpha, \beta, MaximumIterations, projections, chromosomes$
$shortPeriod \leftarrow 1, mediumPeriod \leftarrow 3, longPeriod \leftarrow 5$
$currentSmoothing \leftarrow computeSmoothing(currentLoad, previousSmoothedValue,$
$previousTrend, \alpha)$
$currentTrend \leftarrow computeTrend(currentLoad, previousSmoothedValue,$
$previousTrend, \beta)$
$projections[shortPeriod][shortCurrentLoad] \leftarrow currentLoad$
$projections[shortPeriod][shortPreviousSmoothing] \leftarrow previousSmoothing$
$projections[shortPeriod][shortPreviousTrend] \leftarrow previousTrend$
$projections[mediumPeriod][mediumCurrentLoad] \leftarrow currentLoad$
$projections[mediumPeriod][mediumPreviousSmoothing] \leftarrow previousSmoothing$
$projections[mediumPeriod][mediumPreviousTrend] \leftarrow previousTrend$
$projections[longPeriod][longCurrentLoad] \leftarrow currentLoad$
$projections[longPeriod][longPreviousSmoothing] \leftarrow previousSmoothing$
$projections[longPeriod][longPreviousTrend] \leftarrow previousTrend$
$projections.removeIndex(0)$
$projections.addNew()$
for $i \leftarrow MaximumIterations$ **do**
  $calculateFitness(currentLoad)$
  $doSelection()$
  $findBest()$
  $doCrossover()$
  $doRandomMutation()$
  if $shouldStop()$ **then**
    $break$
  end if
end for
$\alpha, \beta \leftarrow bestCombination()$

The equation shows that the higher the error appears, higher the fitness becomes. As we prefer to have a higher fitness for the chromosomes with the lower error, the fitness is reversed $f = \frac{1}{f}$ after it is calculated.

After calculating the fitness for each chromosome, a natural selection approach is used to calculate the changes of selecting a chromosome. The used selection approach is Roulette Wheel Selection as it considered the fastest among the other selection methods, such as Rank and Tournament approaches. However, it lacks providing more diversified chromosomes. Having more diversified chromosomes allows the process of reaching to an ideal chromosome for our model faster. As we are using a system that is time constrained, the least resource demanding approach was selected.

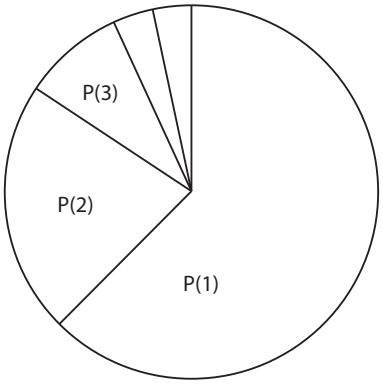


Figure 3.9: Representation of Russian Wheen Selection

The probability of choosing a chromosome using the Russian Wheen Selection is $P(choice = i) = \frac{fitness(i)}{\sum_{j=1}^{n} fitness(j)}$, where $\sum_{i=1}^{n} P(i) = 1$, as shown in 3.9. Two random numbers are generated. The chromosomes that their probabilities contain the random numbers are selected as a candidate parent chromosomes for a child chromosome in the next iteration. The child chromosome is a combination of the the two parent chromosomes, which contains a random number of bits from the first parent and the rest of its size, 12-bits in total, is filled from from the other parent, as shown in 3.10.

After creating a list of all children, a random mutation is applied to the children where a random number is generated at each bit to see if a bit mutation is needed. The bit mutation's goal is to create a diversified child chromosomes. This process happens for a number of iterations. As we are dealing with a real system, an additional measurement is enforced to when the process should exit from the iteration. In the greedy approach, it stops when the goal is reached or no better chromosomes are generated. Waiting until one of the previous statements happen would take a lot of time. Thus, the variant

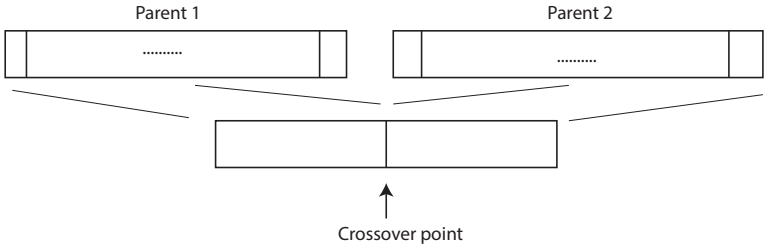


Figure 3.10: GA Crossover

applied a maximum iteration that represents the maximum number of allowed iteration before it stops. After it stops, the best combination of α and β is chosen to be the Holt's parameters in the next balancing cycle.

Algorithm 5 Fitness calculation

for $c \leftarrow NumberOfChromosomes$ **do**
  $\alpha, \beta \leftarrow parseChromosome(chromosome)$
  $smoothness \leftarrow computeSmoothness(projection[0][shortPreviousSmoothness], \alpha)$
  $trend \leftarrow computeTrend(projection[0][shortPreviousTrend], \beta)$
  $SP \leftarrow smoothness + trend$
  $smoothness \leftarrow computeSmoothness(projection[0][mediumPreviousSmoothness], \alpha)$
  $trend \leftarrow computeTrend(projection[0][mediumPreviousTrend], \beta)$
  $MP \leftarrow smoothness + trend$
  $smoothness \leftarrow computeSmoothness(projection[0][longPreviousSmoothness], \alpha)$
  $trend \leftarrow computeTrend(projection[0][longPreviousTrend], \beta)$
  $LP \leftarrow smoothness + trend$
  $f \leftarrow \frac{(|current-SP|)+(|current-MP|)+(|current-LP|)}{3}$
end for

3.3 Evaluation

For the number of variants proposed here, a method of comparison is needed to measure their reliability of giving decent projections. Therefore, an accuracy evaluation is performed to compare their projections with the real readings of loads. In order to get readings of loads, data gathering is established to obtain the needed data for the evaluation. Additionally, this section discusses the time and space complexity of each evaluation to identify the effect of the proposed enhancements and their overhead to the system.

3.3.1 Gathering Data

Different data samples were collected and analyzed in order to have test cases that cover the possible scenarios distributed simulations might experience. These data samples were classified, based on the number of federates, into the following 3 different categories : low, medium, high. In each category, different cases were taken into account, such as actual loads that follow a specific pattern. The goal is to cover as many situations as possible in order to test the close-to-real performance of the different enhancements instead of testing the systems against a specific type of situation. Figure 3.11 shows a snapshot of three data samples of the different federate configurations. For each configuration, a set of 20 data samples were used.

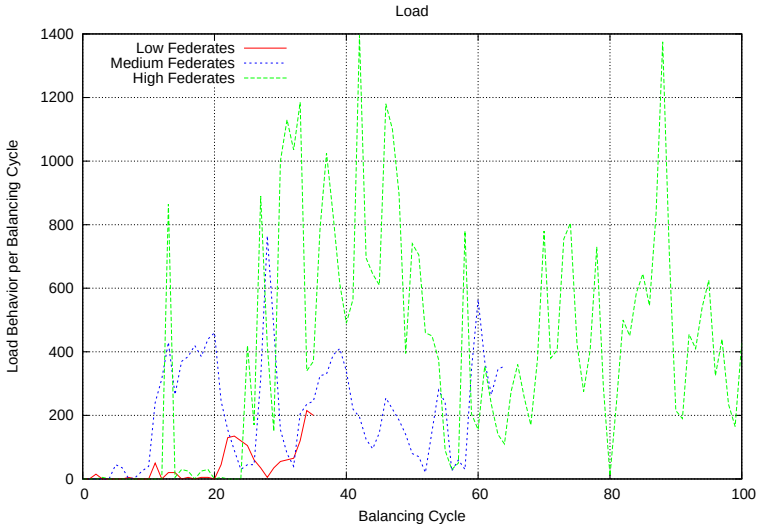


Figure 3.11: Representation of data samples with different setups

3.3.2 Accuracy Comparison

In the evaluation of the proposed approaches, the enhancements are compared against the the prediction model used in the prediction-based distributed load balancing system [49]. The evaluation process is done in two stages. The first stage performs the evaluation of the prediction techniques on data samples, generated by running several distributed simulations on top of a set of shared resources. A systematic computational load was used in the system to emulate simulation overhead. The aim of the first stage is to test the efficiency of the methods and to properly configure them for real simulations. The

second stage is presented in the the Evaluation chapter where one of the variants is implemented on real dynamic load balancing system. In order to compare the performance of the prediction variants, Symmetric Mean Absolute Percentage Error (sMAPE) is used as it is one of the used metrics to measure the prediction accuracy. It represents the average difference between the projection and the actual load over the summation of the projection and the actual, as shown below

$$sMAPE = \frac{1}{n} \sum_{t=1}^{n} \frac{|F_t| - |A_t|}{\frac{|A_t|+|F_t|}{2}} \tag{3.23}$$

First, in order to examine GA's performance, a thorough analysis is established to identify the combination of iterations and chromosomes that fits our case. An accuracy test of different configurations is conducted to compare the error for each computed projection in the different federate setups we have, which is discussed in the previous section.

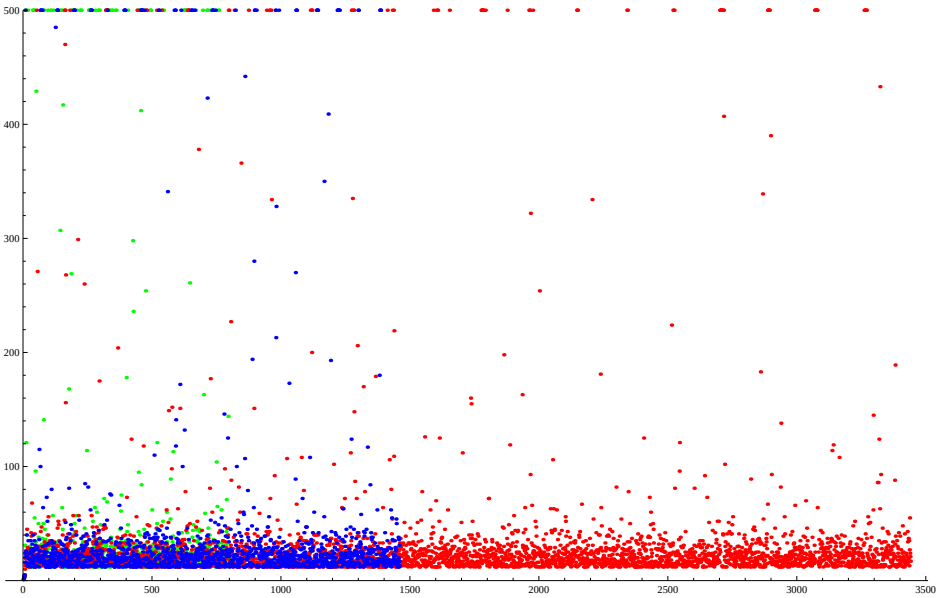


Figure 3.12: Sample Coverage Percentage per # Iterations

There is not any simple and direct rule to define the needed number of chromosomes. However, the more chromosomes we have the more diversified combinations of α and β we get. Therefore, three different numbers (50, 150, and 150) of chromosomes were chosen for the accuracy comparison. On the other hand, finding the suitable number of iterations requires deep analysis to understand the distribution of the iterations. To do

so, the simulation was executed 10 times per each federate configuration with a loose maximum number of iteration that is set to 500. Besides the additional iteration metric, GA still follows the greedy approach and stops when the target is reached (error equals to 0) or the system could not find a better combination of α and β for a consecutive 10 iterations. Figure 3.12 shows the average number of iterations the system needed per each balancing cycle. The least number of iterations needed by the system is 10 and it shows that the system started with a certain chromosome that resulted in the least error between the actual and the projected load for 10 consecutive runs, therefore, it stops and returns that chromosome as the one with the best combination of α and β for the balancing cycle. The maximum, however, is 500 and it is expected as sometimes GA requires more iterations to find the best chromosome, especially when the the error is decreasing in each iteration. The introduced metric enforces the system to stop GA in order to comply with the real-time restriction imposed by the processing time needed per each balancing cycle.

Our initial goal was to start with a maximum number of iterations that could allow finding the suitable α and β for at least 50% of the balancing cycles, without hitting the limits, and then increase the number of iterations to cover more. Figure 3.13 shows the percentage of balancing cycles where the system could find a best chromosome without reaching the limit. For example, GA could find the best chromosome for 64% of the balancing cycles when medium number of federates when the maximum number of iterations is set to 25. Changing the maximum number of iterations to 50 increased the coverage percentage from 64% to 88%. However, increasing the number of chromosomes to more than 50 does not have a noticeable effect on the total coverage

Figure 3.14 shows 2 samples of the average error of the different configuration when SP is calculated with medium number of federates and MP is calculated with high number of federates.

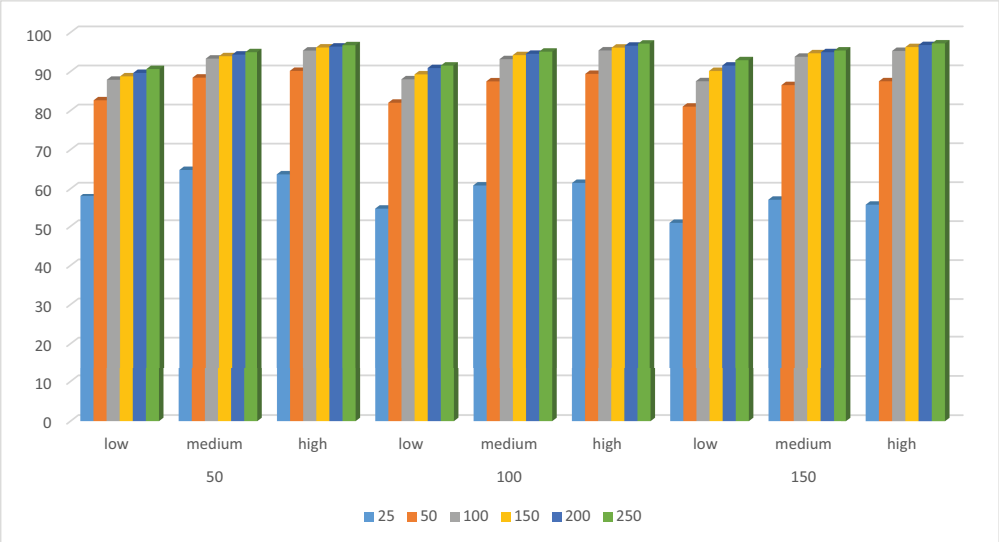


Figure 3.13: Needed iterations per balancing cycle

Because of the importance given to SP calculations, GA tries to provide the lowest error for SP. However, doing so does not always assure the lowest errors for MP. As a result, choosing a high number of chromosomes that run for a long time, long iterations, does not provide any noticeable differences in terms of the total average error. Thus, a number of chromosomes and iterations are chosen so that it provide a decent error with low time and space complexity. By analyzing the different errors for the different case, we chose 50 chromosomes and 50 iterations.

For the other variants, finding α and β that result in a low error is essential. Therefore, a grid search was performed where a set of possible values for α and β, $0 \leq \alpha, \beta \leq 1$,

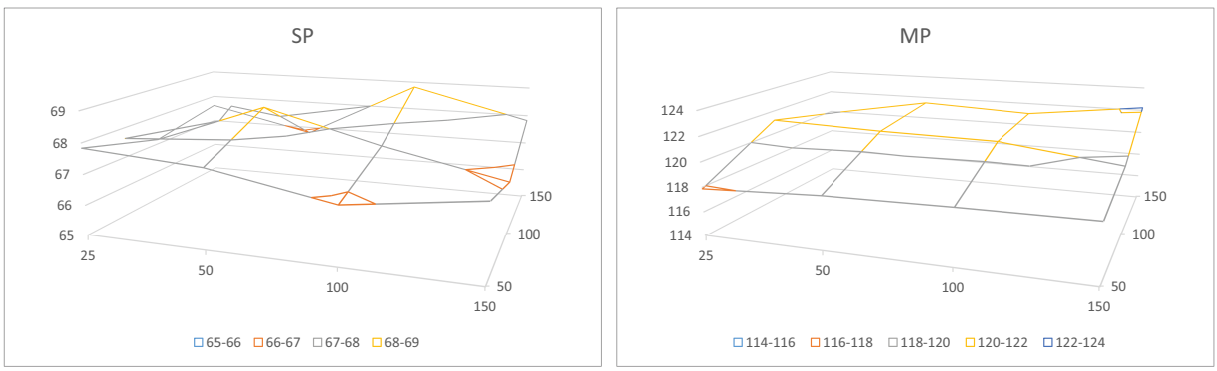


Figure 3.14: The average error per configuration

with a step of 0.01, were thoroughly compared and the combination with the smaller amount of computed error was chosen. The predictions were extensively evaluated and compared using the calculated error by the enhancement. The error may be understood as the difference between the predicted load and the real load value. The average for the errors of each category and each projection was determined, and this is shown in Table 3.1 that lists the rounded average error for Holt's model and its variants.

For the unmodified Holt's model, the grid search returns $\alpha = 0.36$ and $\beta = 0.36$ as a combination with the lowest error. As Cutoff and RT have the characteristics of Holt's model, their parameters were given the same values as of Holt's model.

Table 3.1: Average % Error per variant

	Holt's	Cutoff	RT	SS	SSRT	GA
SP	54	58	40	33	30	65
MP	95	94	90	79	78	113
LP	103	102	100	82	83	120

A set of experiments were conducted to compare the performance of Holt's model and Cutoff. The naive approach (Cutoff) does limit the medium and long term projections as it is supposed to do. Figure 3.3 demonstrates the limiting effect on the long-term prediction. As a result of this limitation on the projections, the Cutoff has set the projections to a threshold instead of to unrealistic projections. This results in a decrease in the error rate of medium and long projections, as shown in Table 3.1. This limitation does not show a noticeable decrease as, from observation, resources usually reach their peak load during the first few balancing cycles which results in turning Cutoff's functionality to a similar functionality of Holt's Model. However, the main task of designing the Cutoff has a negative side-affect on the short term projections. This method decreased the performance of short term projections as a result of applying measures instead of enabling a benefiting from the capabilities of Holt's limiting model.

Reversed-Trend was implemented and evaluated against the data samples. The results, as shown in Table 3.1, show that the Reversed-Trend improves the response speed to load oscillations for short and medium-term projections when compared against Holt's model. Figure 3.4 demonstrates the fast response of RT to oscillations against Holt's model for short-term projections.

A grid search was performed on each system in the Separate System technique. As expected, a low α and β are needed for MP and LP as the dependency on previous loads becomes more important. By analyzing the errors for each combination of α and β, $(\alpha = 0.85, \beta = 0.36)$ provides the lowest error for SP. Similarly, $(\alpha = 0.08, \beta = 0.09)$ and $(\alpha = 0.05, \beta = 0.05)$ return lowest error for MP and LP, consecutively. Table 3.1 shows that by having a separate system for each projection, a higher degree of prediction precision is possible than when calculating the projections with Holt's model system and different time intervals. Figure 3.5 demonstrates the higher amount of precision the Separate System technique offers compared to Holt's model.

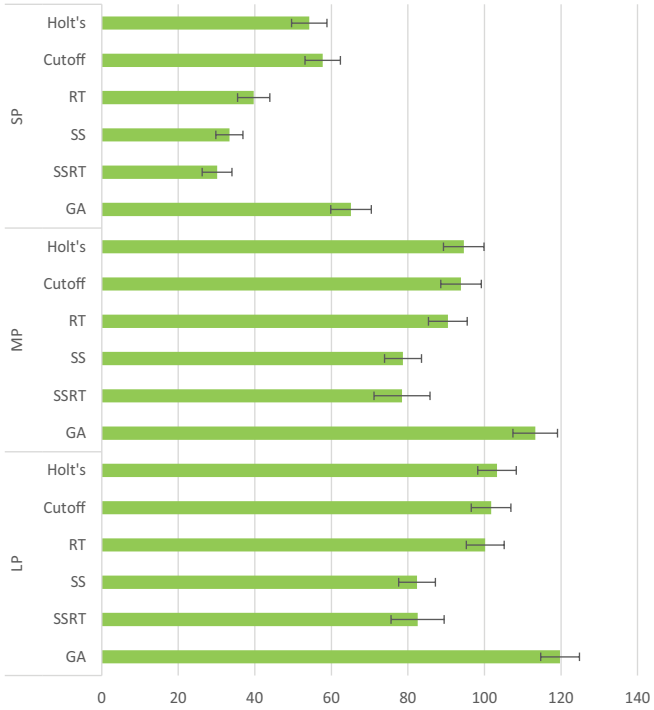


Figure 3.15: Avg Error and 95% confidence interval for each variant

Another variant comes to mind after exploring the different proposed solutions where *each* solve *one* of the previously listed problems. This variant combines the ability of two systems to have a combination that solves both the observed problems with Holt's model. By merging the *Separate Systems* technique with the *Reversed Trend* technique, and by thus generating a *Seperate System Reversed Trend* (SSRT), the performance is expected to improve. RT will add an element of sharpness and quick reaction to sudden

oscillations. Moreover, SS would increase the precision of MP and LP. As expected, Table 3.1 shows that the accuracy of the new variant improved for all projections.

GA's results are a proof that not all extensive computations would give better performance. The concept of using GA matches our requirement in having dynamically adjusted Holt's parameters. However, GA keeps changing them continuously in a way that affects the projections. The main disadvantage of changing the parameters dynamically for a *system that oscillates between statuses* more frequently is that the changes needed to the parameters to let the prediction model fit the new status needs time. Moreover, once a resource's status is changed from stable to oscillating, GA changes Holt's parameters to make Holt's model adjust fast to the change. This sudden change causes unrealistic projections for MP and LP, and indeed this is the main cause of the poor GA performance.

Figure 3.15 summarizes the results of the different Holt variants. Each bar represents the average error of an enhancement with the 95% confidence interval for each projection. Cutoff shows that it is capable of decreasing the error of projections with the limitations it sets to the predicted values. However, the limitations imposed on short term predictions add restrictions to the Holt's model performance which reduces the performance of the short term predictions. On the other hand, RT and its additional sharpness to the forecast affects the performance of long-term predictions. Observing the implementation of RT on different data samples and comparing the results with other models raises the need to have smooth long term predictions instead of sharp predictions for low and medium number of federates. However, the sharpness does add value to the long-term prediction when the number of federates is high and the load of the shared resources oscillate more frequently. The isolated systems in SS improved the predictions as it gives each projection a different treatment for the two parameters of Holt's model. These different parameters show that implementing only one Holt's model is not an ideal solution for data that oscillates frequently. Combining Separate Systems with Reverse Trend shows that the two variants together tackle the drawbacks listed and decrease the total number of error in the projections.

3.3.3 Complexity Comparison

As we are dealing with a balancing system that is time-dependent, a complexity analysis is needed to know which variant fits better than the rest in the deployment in the real system instead of Holt's model. In order to fairly analyze them, only the complexity of

the *prediction model* is presented. As the rest of the balancing system is untouched.

Letting n is the number of resources managed by all CLBs, the number of *times smoothed value and trend* per *balancing cycle* are calculated per is broken down as shown in Table 3.2:

Table 3.2: Time and Space Complexity for the proposed variants

Variant	Time Complexity	Space Complexity
Holt model	$O(2n)$	$O(2n)$
Cut off	$O(6n)$	$O(3n)$
RT	$O(4n)$	$O(2n)$
SS	$O(6n)$	$O(6n)$
SSRT	$O(12n)$	$O(6n)$
GA	$O(35050n)$	$O(100n)$

Holt's model calculates each of the *smoothed value* and *trend* once. Based on the two values, the projections are computed. However, we are not interested in the projections' calculations as computing their values is something essential and is done no matter what variant we use. So, for Holt's model, there are only two calculations, one for the smoothed value and the second is for the trend. Thus, the complexity is $O(2)$ per resource. As we have n resource, the total added time complexity is $O(2n)$. For space complexity, Holt's model require 2 spaces in memory, one for the previous smoothed value and the second is for the previous computed trend. Thus, the space complexity is $O(2n)$.

Cutoff performs the same calculation as Holt's model. However, Cutoff performs an extra 4 operations per balancing cycle: comparing the threshold against SP, comparing the threshold against MP, comparing the threshold against LP, and setting the threshold. Thus, per balancing cycle, Cutoff performs $O(6n)$ for the. Cutoff requires one more space in the memory to store the maximum threshold. Adding that space to the space needed for the previous smoothed value and the trend results in $O(3n)$ space complexity per resource.

RT reverses the sign of the tendencies when they are different. This is done through a comparison. Moreover, RT does not reverse until a certain measurements are followed: no sign reversing in case of small load oscillations. Thus, the complexity is $O(4n)$. However, it does not need any more space in memory to perform its task. Thus, the space complexity is $O(2n)$.

For SS, there exists a different prediction model to calculate each projection. The prediction model it uses are based on Holt's mode, resulting in a complexity of $O(3 \times 2n) = O(6n)$. However, in this case the space complexity is higher than the others. The reason is that we need to keep different historical smoothed values and trends in memory. As the space complexity for Holt's is $O(2n)$, where it needs a space for the previous smoothed value and another space in memory for the trend, SS would need three times the space complexity of Holt's model. The space complexity for SS is $O(6n)$.

SSRT combines SS with RT. Thus, the complexity is expected to be higher as the number of operations performed is bigger now. SS part in SSRT performs $O(6n)$ to compute the projections. However, RT plays a rule as well in this variant. When adding the statements of reversing trend to the SS, we would end up with $O(6 \times 2n) = O(12n)$. The space complexity, however, is the same as SS as RT does not have any space requirements.

GA is considered the worst in both complexities. The used GA model uses 50 chromosomes, contains the details of α and β, in 50 iterations. For each configuration, the fitness is calculated by computing the smoothed value and the trend. The smoothed value is computed $50 \times 50 = 2500$ times, the same is applied to the trend. it is true that it seems bad in terms of time complexity, however, there is another complex method in GA that produces much more complexity and that is the mutation function. Mutation has to go through the children of each iterations to perform bit mutation. For its worst case, it has to go through each bit in each chromosome, in every iteration. Given we have 50 chromosomes, 50 iterations, and 12 bit per chromosome, mutation needs $50 * 50 * 12 = 30000$ per resource. It is a demanding approach, however, there are other factors that are not considered such as defining the children for the next cycle, once optimized, it can be reduced to $O(chromosomes) = O(50)$. We can say that GA is a demanding approach in terms of computing power. GA requires more space to perform its functionality. GA needs 50 spaces to save the chromosome's information. Moreover, it needs another 50 to save its children information before starting the next cycle.

3.4 Summary

The different variants solve the weaknesses identified when using Holt's model in similar environments. The proposed methods solve a weakness using its own way. Cutoff limits the unrealistic projections by applying a threshold. RT changes the trend in a way to add more sensitivity to the system in term of detecting load oscillations. SS assigns a different prediction model for each projection to reduce the unreasonable projections.

GA tries to solve the detection of oscillations faster, however, it failed to provide decent projections.

Based on the accuracy comparison and the complexity analysis, it is clear that GA is the worst option someone can pick for such cases. SSRT provided the least error to complexity ratio. Therefore, SSRT is the variant that will be implemented in the dynamic load balancing system to evaluate its performance against Holt's model.

Chapter 4

Federate Migration Decision-Making Approaches

The incorporated dynamic load balancing system goes over two different phases in order to perform federate migrations in a third phase to balance the shared resources the simulation runs on. The first phase makes use of the prediction model, along with the prediction history, to compute projections. A number of variants of Holt's model were discussed in the previous chapter. The second phase of the system is the pair matching between resources to check if there is a need to trigger a migration. This chapter discusses the development of different and more dynamic pair matching approaches. A deep analysis that is performed on the internal structure of the dynamic load balancing system shows a necessity to develop such approaches. The original systems' approach to trigger migration was not dynamic in a fashion where it can adjust itself to other prediction models or variants. Algorithms 2 and 3 show that the adjustment of the values of min, θ, and δ are based on the direction (tendency) of the source's load, the direction of the destination's load, and the projection type (SP, MP, or LP). As the system uses Holt's model in the prediction engine, the parameters (min, θ, δ) were tuned up, based on predefined values, to trigger a low number of migrations. The combination of these parameters produce two different thresholds that once exceeded, the system sees a justification to trigger a federate migration to balance the load of the shared resources.

4.1 Proposed Solutions

This section discusses the logic behind the proposed federate migration decision making approaches. The proposed threshold-independent approaches introduce concepts that aim to better understand the behavior of the resources in a dynamic fashion to decide if a migration is needed, rather than using thresholds.

4.1.1 Restricted Approach

The Restricted Approach (RA) replaces the dependency on the predefined thresholds with a rule-based Expert system. The proposed approach depends on metrics that are used in the original federate migration process along other factors. RA uses the load and direction of the source and destination nodes, the mean of all nodes in the current balancing cycle, and a computed tolerance. The tolerance is a value proposed in the original system that defines a range that once used along the mean, the dynamic load balancing system can define a *balanced* area where resources with loads that reside in that area are considered balanced. RA uses the concept of tolerance to compute a region around the quantitative value of the resource's load to define a *resource area*.

As RA is an expert system, a set of rules are defined based on the observation of when the migrations should be triggered. The rules do not depend on any predefined value and are based on whether there exists an intersection between the *balanced area* and the *resource area*. The existing of such intersection raises a *balanced flag* as the resource is near the balanced region and there is no justification to have it involved in a federate migration. Figure 4.1 demonstrates the balanced and the resource areas. A resource is considered *safe* when its resource area does not intersect with the balanced area. Once there is an intersection, the resource is considered *not safe* and the dynamic load balancing system would include the not safe resource in its load balancing process.

Figure 4.2 shows the flow chart RA goes through to see of there is a need to trigger a federate migration. The flow chart shows that RA is a loose approach as almost all of its covered scenarios end up in a migration, which eventually would lead to issuing a big number of migrations. Because of how the incorporated dynamic load balancing system works, once a migration is triggered, the two involved resources are removed from the candidate list. Once the balancing system analyzes the list of migrations needed based on short term projections, a shorter list of candidates is sent for further processing to analyze possible federate migrations that is based on medium term projections. The same process is applied when the list is sent for long term projection processing.

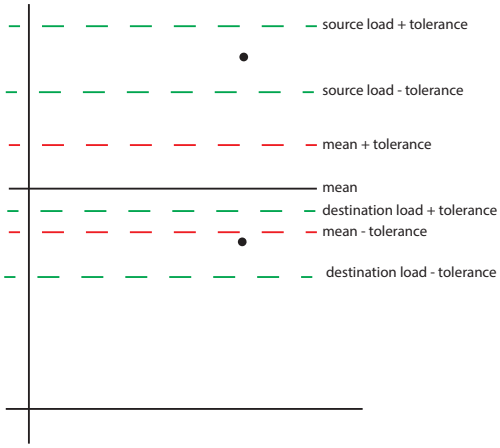


Figure 4.1: RA Areas

Based on the flow chart in Figure 4.2, there is only one case where the load balancing system does not migrate federates. The orphan case is when both resources are not safe, the destination has an increasing load tendency, and the source has either a decreasing or stable load tendency. In simple words, both resources are heading towards the balanced area. Thus, issuing a federate migration from the overloaded source to the underloaded destination would end up in the possibility of changing the destination's status from underloaded to overloaded in the few next balancing cycles. Additionally, the previous rule prevents issuing migration calls in the future when overloaded resources become underloaded resources.

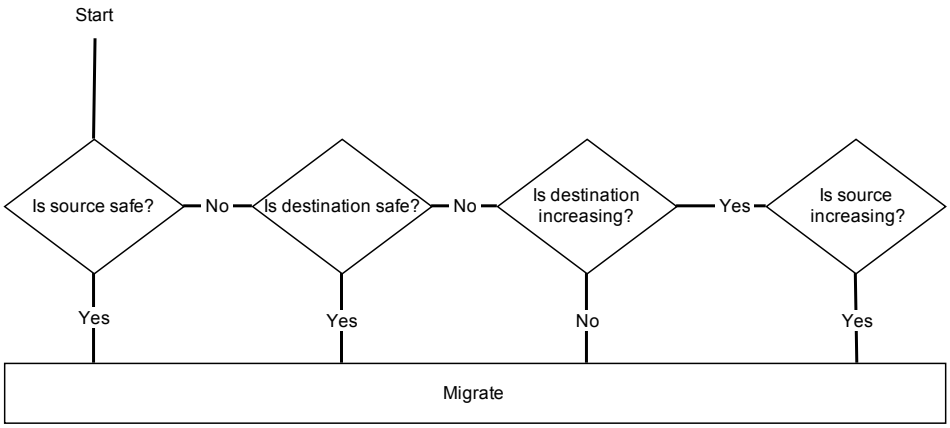


Figure 4.2: RA flow chart

4.1.2 Direction Ratio Approach

RA uses a loose set of rules that can cause inconsistency in the federate migration process. Thus, a more restricted approach was needed. The restrictions to be applied in the new approach must limit the number of migrations to only those that would eventually lead to a balanced environment. In order to do that, Direction Ratio (DR) adds *Ratio* to the list of metrics used by RA. The ratio represents the percentage of source's load tendency to the destination's load tendency. Unlike RA, DR makes use of the directions in most of its rules.

In addition to RA rules, DR applies more limitations to the number of triggered migrations by analyzing the direction of the candidates' loads. Examining the different possibilities of the directions gives an insight on whether a migration has the possibility of affecting the load and status of both candidates. The ratio is used as a metric to calculate the effect factor. The ratio is then compared to a fixed value, and based on the comparison, DR either triggers or ignores the migration. Figure 4.3 shows the flow chart of DR and how the ratio is used to filter the list of migrations. For example, having a scenario similar to 4.4 where

- Source is not safe and has an increasing load tendency
- Destination is safe and has an increasing load tendency but slower than the source
- Ratio > 0.5

Based on observations and analysis, the previous scenario requires a federate migration to lower the increasing load tendency of the source and make it stabilize before transferring another federates in a future balancing cycle. Moreover, the load transfer would make the convergence process faster by increasing the load tendency of the underloaded and decreasing the load tendency of the overloaded. However, for a similar example but where the ratio is less than 0.5, this means that the underloaded resource has an increasing load that moves fast towards the balanced region. Regardless of migrating a federate from the overloaded resource would decrease its load's tendency, transferring a load to an underloaded resource that is heading fast to the balanced area is not wise as it has a higher chance of becoming overlaoded in a future balancing cycle. Thus, no load migration is triggered. Afterwards, the system would use the DR approach to decide if a migration is needed between the same overloaded source a and another underloaded resource.

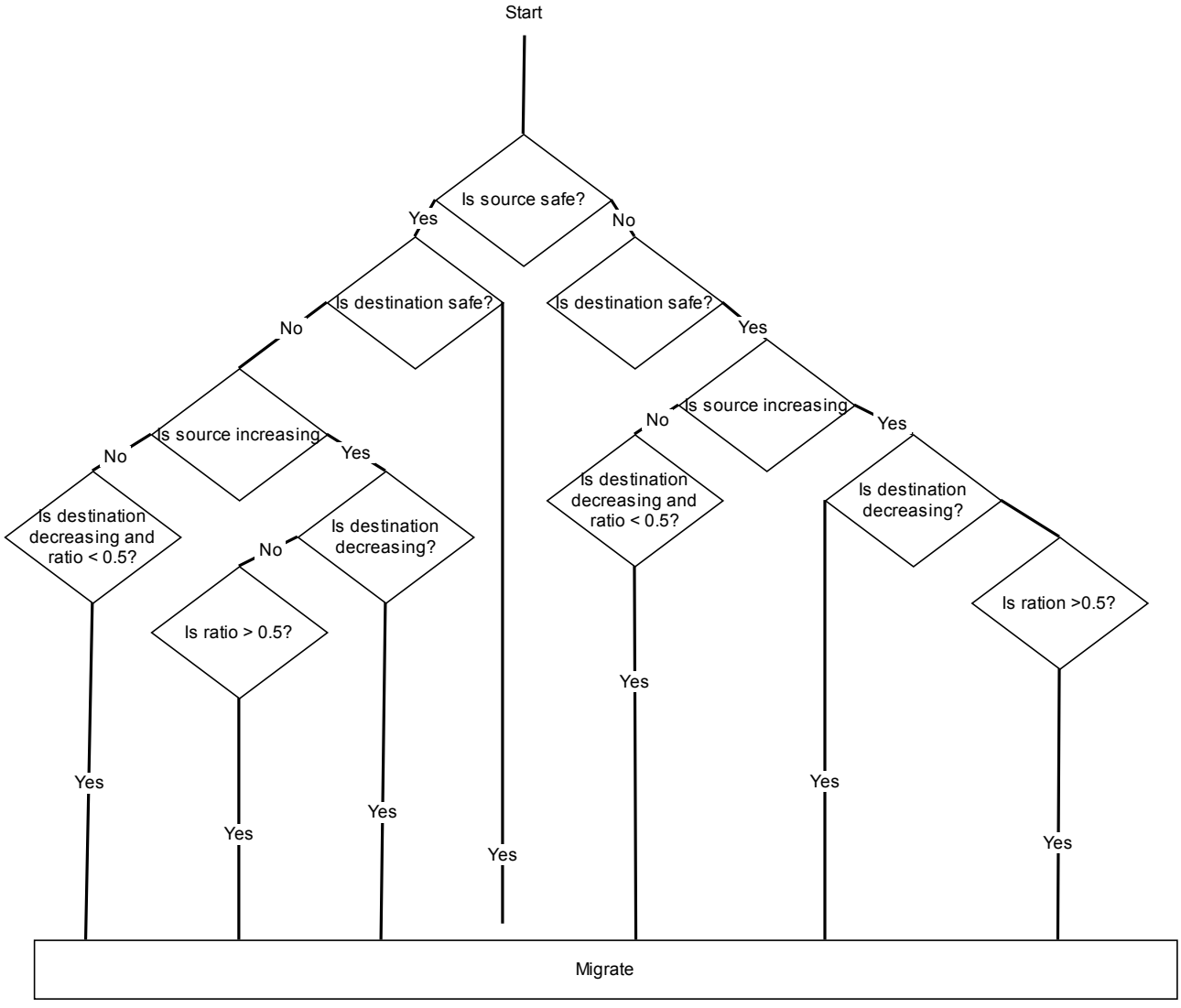


Figure 4.3: DR Flow Chart

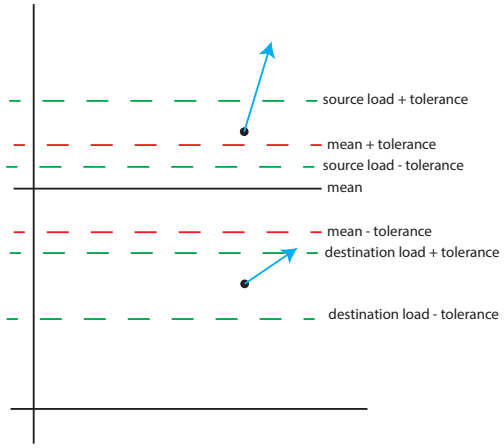


Figure 4.4: DR Flow Chart Example

4.1.3 Fuzzy Approach

A different idea was desired to increase the balancing convergence speed of the shared resources. This led to start investing a new approach that identifies different kinds of loads and reacts differently for each type of load. The idea is based on categorizing the loads into sets that represent the level of load the resource is under. Instead of a federate migration decision-making approach that depends on comparing different metrics, the desired approach would make use of the resources' categories to trigger a number of federates to insure a quick convergence speed. Fuzzy Logic is an approach that works by categorizing inputs and provides an output based on predefined rules. Fuzzy Approach (FA) assigns each load to a set, then goes over a number of predefined rules to see which one is applied to the current situation and then perform the rule's actions.

The approach has 6 predefined sets, where a pair of 3 sets are dedicated to cover each status, overloaded and underloaded. The 3 sets per status covers different ranges of the loads: *extreme*, *normal*, and *low*. The combination of the status with their ranges of loads results in the following fuzzy sets: *extremely overloaded*, *overloaded*, *slightly overloaded*, *slightly underloaded*, *underloaded*, *extremely underloaded*.

Based on these sets, the following rules are introduced to achieve the goal of fast balancing convergence. This is reached through migrating more that one federate in case there is a big gap between the overloaded and the underloaded and one federate if the gap between the candidates are small.

1. Source is *extremely overloaded* and destination is *extremely underloaded*, then mi-

grate 2 federates

2. Source is *extremely overloaded* and destination is *underloaded*, then migrate 1 federate
3. Source is *extremely overloaded* and destination is *slightly underloaded*, then do not migrate any federate
4. Source is *overloaded* and destination is *extremely underloaded*, then migrate 1 federate
5. Source is *overloaded* and destination is *underloaded*, then migrate 1 federate
6. Source is *overloaded* and destination is *slightly underloaded*, then do not migrate any federate
7. Source is *slightly overloaded* and destination is *extremely underloaded*, then migrate 1 federate
8. Source is *slightly overloaded* and destination is *underloaded*, then migrate 1 federate
9. Source is *slightly overloaded* and destination is *slightly underloaded*, then do not migrate any federate

At each balancing cycle, FA creates 3 equal sets for each status that ranges from the min/max to the balanced region, as shown in Figure 4.5. A loop starts by selecting candidates and assign each of the candidates to their respective category. FA then evaluates the appropriate rules. Once a rule is evaluated, the system will transfer the proposed number of federates by FA from the source to the destination.

4.1.4 Impact Approach

All of the proposed approaches, including the approach that is used in the original system, remove the the candidates from the candidate list once a migration is triggered. When the resources are removed from the candidate list, the balancing system would not consider analyzing any possible load migrations in the current balancing cycle or the next one for the removed candidates. Because of this behavior, the balancing system does not take into account any other possible scenarios, that involve the removed resources, which could be more beneficial to the shared resources. Thus, Impact Approach (IA) is designed to propose analyzing possible migrations and issue the most beneficial ones.

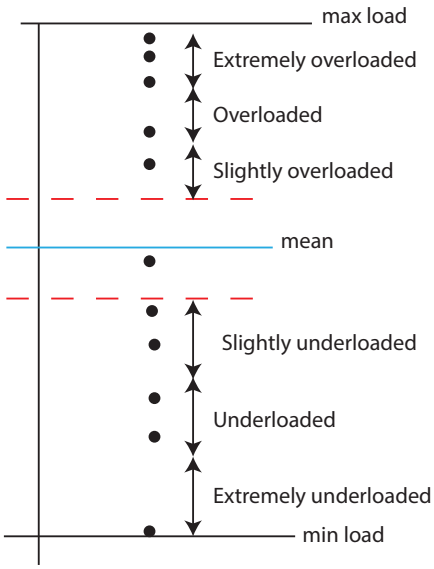


Figure 4.5: Fuzzy Approach: Dividing Sets

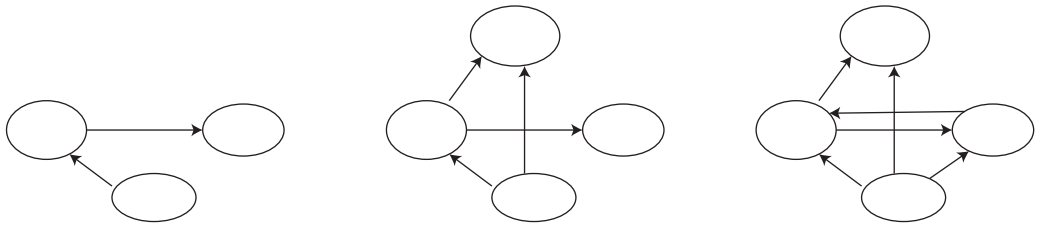
Impact Approach (IA) takes into account all possible scenarios any resource is involved in and triggers the most valuable and set of migrations. In order to do so, IA computes an *impact factor* for each possible load migration, represented by

$$Impact = \frac{srcLoad - mean}{dstLoad} \tag{4.1}$$

The mean is subtracted from the the load of the source to bring the source's load closer to the destination's load. The impact factor's value represents the degree of impact of the migration to the shared environment. The higher the value is, the more important it is to trigger this migration.

For this approach to work, it first goes over all the possible migrations for the short term projections. The possible migrations are presented to the system as a unidirectional graph, from an overloaded source node to an underloaded destination node, and the impact factor as the weight of the edge. Then, the same candidate list is sent to be processed for medium term projections. The possible list of migrations is added to the unidirectional graph. When IA finishes analyzing the possible load migrations for the medium term projections, the candidate list is sent untouched to be evaluated for long term projections. Similarly, the list of migrations are added to the unidirectional graph.

Once all possible migrations are computed, IA goes over through the graph and first perform the migration with the highest impact factor. Once it issues a high impact



(a) SP migrations (b) MP migrations (c) LP migrations

Figure 4.6: IA migrations

migration, IA removes the candidates of the involved migration and their respective edges from the graph. The process continues until no more edges in the graph. The remaining resources are nominated as candidates for the next balancing cycle.

4.2 Evaluation

For the evaluation part, we needed a quick method to analyze the behavior of the proposed federate migration decision-making approaches. Using real clusters to run real simulations for tuning up and further analysis would end up in taking months before the results are out. Thus, an emulator was built which is a ported version of the dynamic load balancing system that works on offline data samples instead online data. However, the offline data was acquired by running real distributed simulations without balancing capabilities. The data samples were extracted from running real simulations for 10 runs on the same setup used in the previous chapter. The results shown below *do not* represent the actual number of migrations nor provide an estimation. The emulator was used to study the *behavior* rather than *performance*. Examining the behavior allowed modifying the approaches in a better manner.

4.2.1 Triggered Migrations

The behavior to be examined is the distribution of the migrations triggered per each projection in a single balancing cycle. Figure 4.7 shows the behavior for such output per each of the proposed federate migration decision-making approach.

By looking into the total amount of migrations, the approaches have the same behavior. However, FL exceeds the other approaches. The reason of the high number of migrations lays in how FL categorizes the candidates. When FL categorizes the over-

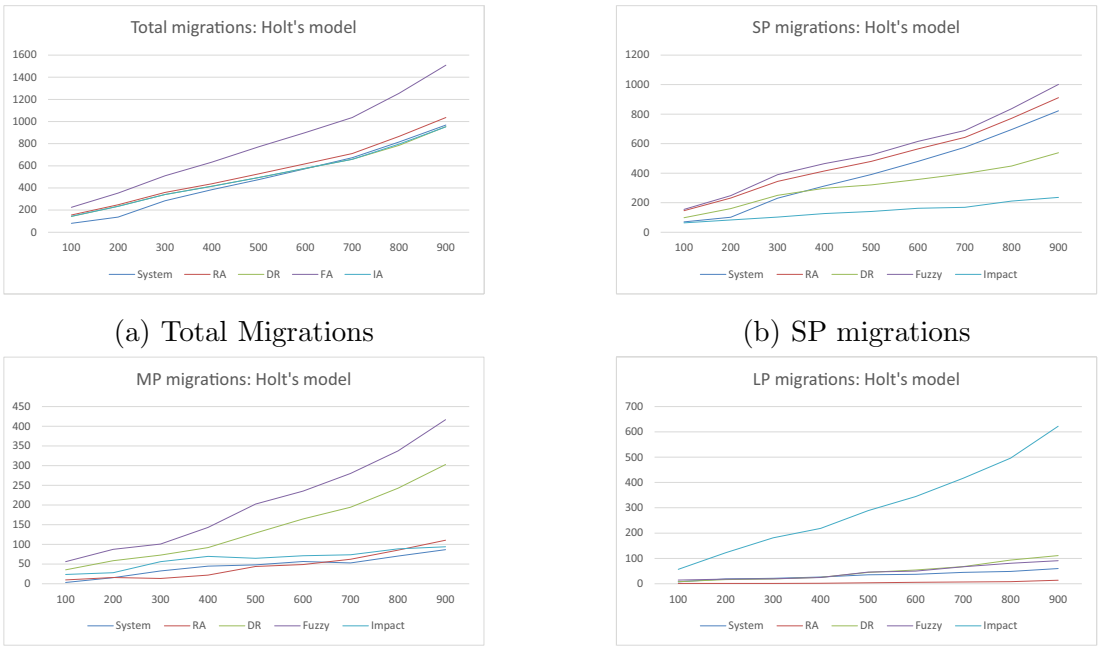
loaded resource as an *extremely overloaded* and the underloaded resource as an *extremely underloaded*, FL transfers two federates from the overloaded to the underloaded instead on one. The previous rule does play an important rule towards increasing the convergence speed to have a balanced environment. However, its drawback is that it triggers more migrations.

This rule is particularly applied the most in these results due to the way FL is implemented. FL creates *always* the *extremely overloaded* and *extremely underloaded* that are based on the *current* loads instead of the *overall* load of the system. In a nutshell, FL creates the extremely overloaded set even when the load is not considered extremely overloaded for the shared resources as a whole, however, it represents an extremely overloaded load for the list of loads in the current balancing cycle. To confirm our doubts, we ran experiments to extract the number of times each rule is applied. Figure 4.8 is an example of the number of hits per rule for two different federate frequencies. For all of the series, data samples, the hits for the first rule is the most. As the first rule triggers two migrations, this justifies the high number of migrations and confirms our doubts that the first rule is the cause behind the high number of migrations for FL. The future work details a draft on how this would be solved and improved.

To better understand the behavior of the proposed approaches, the numbers of migrations triggered for SP, MP and LP were analyzed. By comparing the results of the approaches with the results of the original approach (System), RA produces more SP migrations. This is justified by the loose rules we have defined in RA. The more loose the rules are, the more migrations are to be triggered. Because of the high number of migrations initiated in SP phase, the less candidates are sent to be processed in MP phase. The effect of having low number of candidates is seen in the MP migrations of RA compared to the original where the number of migrations in RA is less than the original system. However, running simulations with more federates results in more inconsistencies, where the looseness is taken advantage from and RA starts producing more migrations than the original system. After this stage, RA has consumed most of its candidates results in low migrations initiated in LP.

However, DR applied more restrictions and this resulted in reducing the number of migrations in SP. This allows for more candidates to be sent for MP phase. At this stage, DR has provided MP phase with the highest number of candidates, therefore, more MP migrations than the original system and RA. Eventually, sending more candidates to LP.

IA shows the lowest number of migration triggered in SP. By looking into Figure 3.1, we can see that the SP has the lowest projections among the rest (MP and LP). As a



(a) Total Migrations

(b) SP migrations

(c) MP migrations

(d) LP migrations

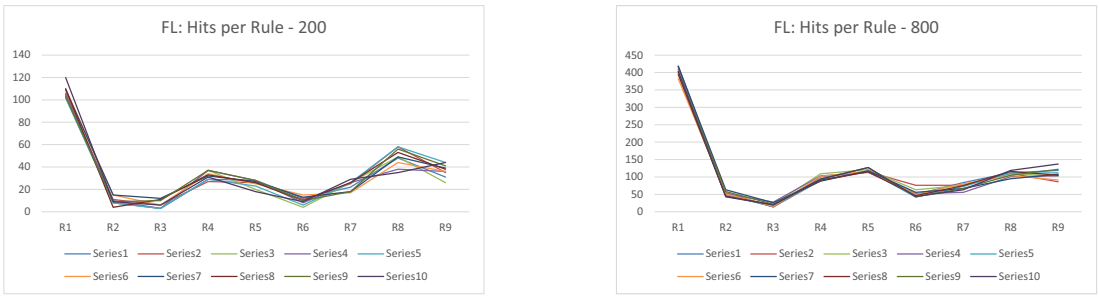
Figure 4.7: Migrations per approach

result, The impact factor computed for SP is smallest and LP is the highest. From how the IA works, IA gives more importance to migrations with the highest impact factor first. Therefore, the number of migrations triggered in LP is the highest, followed by MP, and then SP. This shows that the migrations are issued in a manner where LP migrations are given more importance than MP, which is higher than SP.

Because of the implementation of the rule where two federates are transferred, FL produces the highest number of migrations in SP. It is true that the number of candidates sent for MP processing is low at this stage, however, the rule once again is applied, at least once, which results in a high number of migrations in MP. At LP stage and because most of the rules initiate a migration, the number of candidates is lower than the rest. The two-migration rule again plays an important role here as it increases the number of initiated migrations.

4.2.2 Complexity Comparison

Once again, and similarly to the variants, a complexity analysis is performed to get an idea of the performance of each approach. However, due to the implementation



(a) Hits per rule for 200 federates

(b) Hits per rule for 800 federates

Figure 4.8: FL: Hits per rule

complexity of the approaches, only the most important and distinguished time consuming functionalities, other than the ones shared with the original system, are listed below.

In terms of processing, the approach used in the original system, RA, and DR have the exact structure and all of the major functionality are exactly the same. They are different in the conditional statement the system go through to see if a migration is needed. Therefore, RA and DR do not introduce any major overhead.

FL follows every step in the original system. However, it goes over two additional phases that introduce a slight overhead. FL needs to perform a one time $O(n)$, n is the number of shared resources, evaluation to go over all loads to generate the fuzzy sets. Once the source and the destination are assigned to sets, FL needs $O(9)$ per node to go over the rules that define the needed number of migrations, which results in $O(9n)$. These two phases generate an overhead with a time complexity of $O(n + 9n) = O(10n)$ per balancing cycle. in terms of storage complexity, the extra space needed by FL is a 6-long array that holds the value of which each range ends. This storage is shared among all resources, concluding an extra $O(6)$ storage needed per balancing cycle.

IA is the most complex and demanding approach. The first difference noticed in IA is the way it handles triggering migration. IA considers all the migration possibilities and nominates a migration based on the computed impact factor. The worst case in such scenario is when the resources are evenly divided into overloaded and underloaded. As each overloaded resource will have an edge to connect to an underloaded resource, the maximum number of possible migrations is $O((\frac{n}{2})^2)$ as each resource will have a connection to at most half of the resources. This process is initiated three times, once per each projection phase (SP, MP, and LP). Thus, the complexity added just by analyzing the migrations is $O(3\times(\frac{n}{2})^2)$. However, the same time complexity is needed for calculating the impact factor. Therefore, the complexity after computing the impact factor is $2\times$

$O(3 \times (\frac{n}{2})^2) = O(6 \times (\frac{n}{2})^2)$. This is not the most extensive part of IA. After a list of all possible migrations are generated, IA goes over all of the edges to find the maximum impact to initiate its respective migration. The search complexity for the maximum impact is the same size of the list of all possible migrations, $O((\frac{n}{2})^2)$. This is, however, is only for the first search. After the maximum impact is found and a migration is initiated, the migration is deleted from the migration list. The deleting process can be combined with the search for the next highest impact to save us another search, which costs $O((\frac{n}{2})^2)$. Once the migration is removed, the graph will be smaller as two nodes and an edge are deleted. For the search to start again, it needs $O((\frac{n-2}{2})^2)$ to finish as two nodes are deleted. We can see a pattern in the search complexity, it starts with $O((\frac{n}{2})^2)$ and then $O((\frac{n-2}{2})^2)$, we can see that once another migration is initiated, two nodes are removed and the next search complexity is going to be $O((\frac{n-4}{2})^2)$. This process goes until all nodes are removed. As we are removing a pair of nodes at a time, this means the process of deleting is going to continue $\frac{n}{2}$ times, resulting in a search complexity

$$3 \times \sum_{i=0}^{\frac{n}{2}} (\frac{n-2i}{2})^2 = \frac{1}{32} n(n+2)(7n+1) \tag{4.2}$$

The summation is multiplied by three because the calculations we did are only for finding migrations for 1 projection. As the system generates the possible set of migrations for three different projections, the summation is multiplied by 3.

In terms of space complexity, each possible migration needs to have 3 spaces to be represented: a space for the source node, another space for the destination node, and the last space for the impact. Based on the previous time complexity, the total number of migrations for the three projections is $O(3 \times (\frac{n}{2})^2)$, this results in a $3 \times O(3 \times (\frac{n}{2})^2) = O(9 \times (\frac{n}{2})^2)$.

4.3 Summary

A number of federate migration decision-making approaches are proposed aiming to work along the previously developed variants. The proposed approaches in this chapter are important to make the system adaptive to our changes to Holt's model. The approaches remove the dependency on the predefined thresholds that would not make use of the full potential of our variants when applied in a real dynamic load balancing system

RA and DR introduce rules that are independent to any fixed value. Both approaches cover different set of rules and present different flexibilities. FL assigns resources to sets

and compares the combination against predefined rules to trigger the required number of migrations needed for the system to converge faster. IA shows a flexible approach to analyze all possible migrations and trigger the most important ones first through the use of impact factor.

4.4 Experimental Results

In these sets of experimental results, the use of Holt's model and SSRT, as prediction models, is combined with each of the federate migrations decision-making approaches. The goal is to analyze the gained performance of the dynamic load balancing system for each combination. Moreover, this analysis would help identifying the suitable use cases for the different combinations.

The testbed for the experiments was a heterogeneous environment that consisted of federates deployed in the following two clusters of computing servers: IBM and Dell. The IBM cluster was composed of 23 computing servers interconnected by a gigabit Ethernet network; each server contains a Core 2 Duo 3.4 GHz Intel Xeon CPU and 2GB of DIMM DDR RAM. The Dell cluster contains 16 computing servers interconnected through a Myrinet optical network that allows for data transmission up to 2 gigabits per second; also each computing server contains a Quadcore 2.40GHz Intel Xeon CPU and 8GB of DIMM DDR RAM. The two clusters are interconnected through their management nodes with a Fast-Ethernet link. Linux operating system was installed in every server. Globus Toolkit 4.2.1 was set up to support the balancing systems; and, HLA platform with RTI version 1.3 was used to coordinate the experimental simulations.

For the balancing systems evaluated in this experiment, their balancing elements were deployed on all computing servers, and the CLBs were placed on each cluster management server. The baseline, as well as the initial configuration of each simulation, consisted in evenly deployed federates on the 39 shared resources and one of the servers was dedicated to running the HLA RTI executive. In the simulation scenario, 1 to 900 federates controlled the movement of objects (tanks) during 100 time steps. Limited to the communication influence on the simulations, each federate managed only one object and calculated the movement of its tank through highly intensive computational tasks.

The following discussion is based on the results seen in Figures 4.9, 4.10, and 4.11 and Tables A.1, and A.2. Overall, SSRT-System performed the best in terms of execution time with slightly less number of migrations when compared with Holt-System. For a method to perform well and provide a better execution time, the method needs to

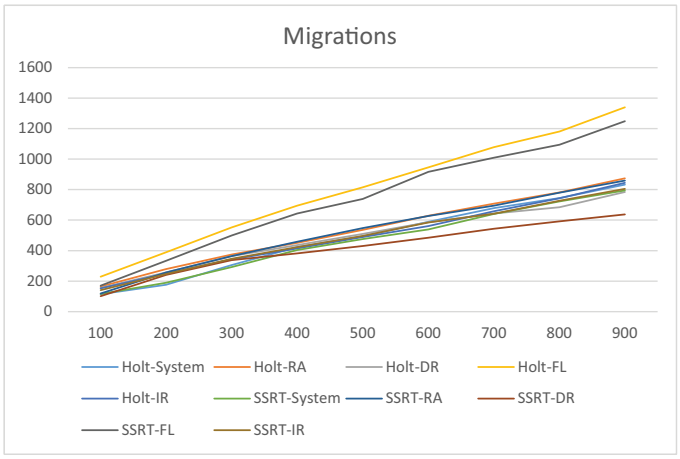


Figure 4.9: Migrations per number of federates

utilize the more powerful resources to compute the distributed simulations. This requires issuing migration calls from a resource with low specifications to a resource with higher specifications. As our infrastructure groups resources with similar specifications into one cluster, issued migrations have to go from one cluster to another which requires initiating *external migration calls*. By looking into Figure 4.11, SSRT-System is the method with the highest number of external migrations. This is a result of enforcing thresholds by the original system on SSRT. These enforcements reduced the number of the triggered migrations to transfer federates within the cluster, therefore, allowing more migrations to be nominated in the process of selecting migrations between clusters.

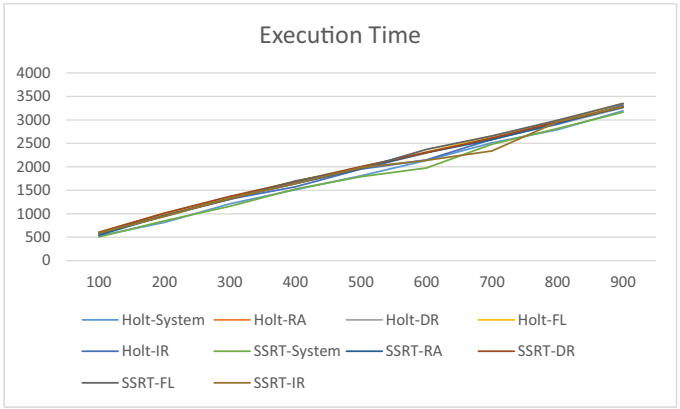


Figure 4.10: Execution time per number of federates

Holt-System comes right after to SSRT-System in having a low execution time. As detailed in SSRT-System, Holt-System applies thresholds as a measurement to trigger a migration. The thresholds' effect on the number of internal migrations decreases as the differences between the load of the overloaded source and the load of the underloaded destination increases. This happens when the computing nodes process a simulation with high number of federates. In our case, the more federates there are in the simulation, the more migrations are triggered for interdomain, within cluster, load migrations.

Because the unrealistic projections of Holt's model can overcome DR's rules, Holt-DR execute the distributed simulation with an increase up to %19 when compared to SSRT-System. Holt's model produces high predictions and computed trends. Thus, the effect of the ratio becomes week in employing restrictions to decrease the number of migrations. However, DR plays an important rule as it chooses the most suitable migrations rather than triggering a less important migrations.

As RA has loose rules, RA-dependent systems trigger more migrations than needed within its resources. These migrations lower interdomain migrations which increase the performance and reduce execution time. Thus, resulting in a bad performance, up to %23 increase in execution time compared to SSRT-System. The reason why SSRT-RA is better than Holt-RA relies in the precision of the projections. As a result, SSRT have a better picture of the projected load of each resource and wisely performs migrations accordingly.

Implementing SSRT-DR results in slow execution time with an increase up to %24 in the execution time when compared to SSRT-System. However, as DR examines the migrations and triggers the most valuable ones, SSRT-DR performs %24 less migrations while mostly are internal migrations.

Because IA takes into account all possible migrations and then goes over them to issue migration calls, the number of migrations is not suitable for simulations that require communication more than computation. The number of migrations can go up 51% when compared to other configurations. As IA gives more importance to the migrations within a cluster, the execution time is not as good as configurations that focus on inter-domain migrations. As a result, IA shows an increase, up to, 16% in execution time.

FL shows an increase up to 227% in the number of migrations as it migrates, at least once, two federates per balancing cycle which is the result of defining the candidate resources as *extremely overloaded* and *extremely underloaded*. This increase in migration calls results in a slow execution time, at most 22% increase. SSRT part in SSRT-FL has an positive effect on the results when compared to Holt-FL as it provides better

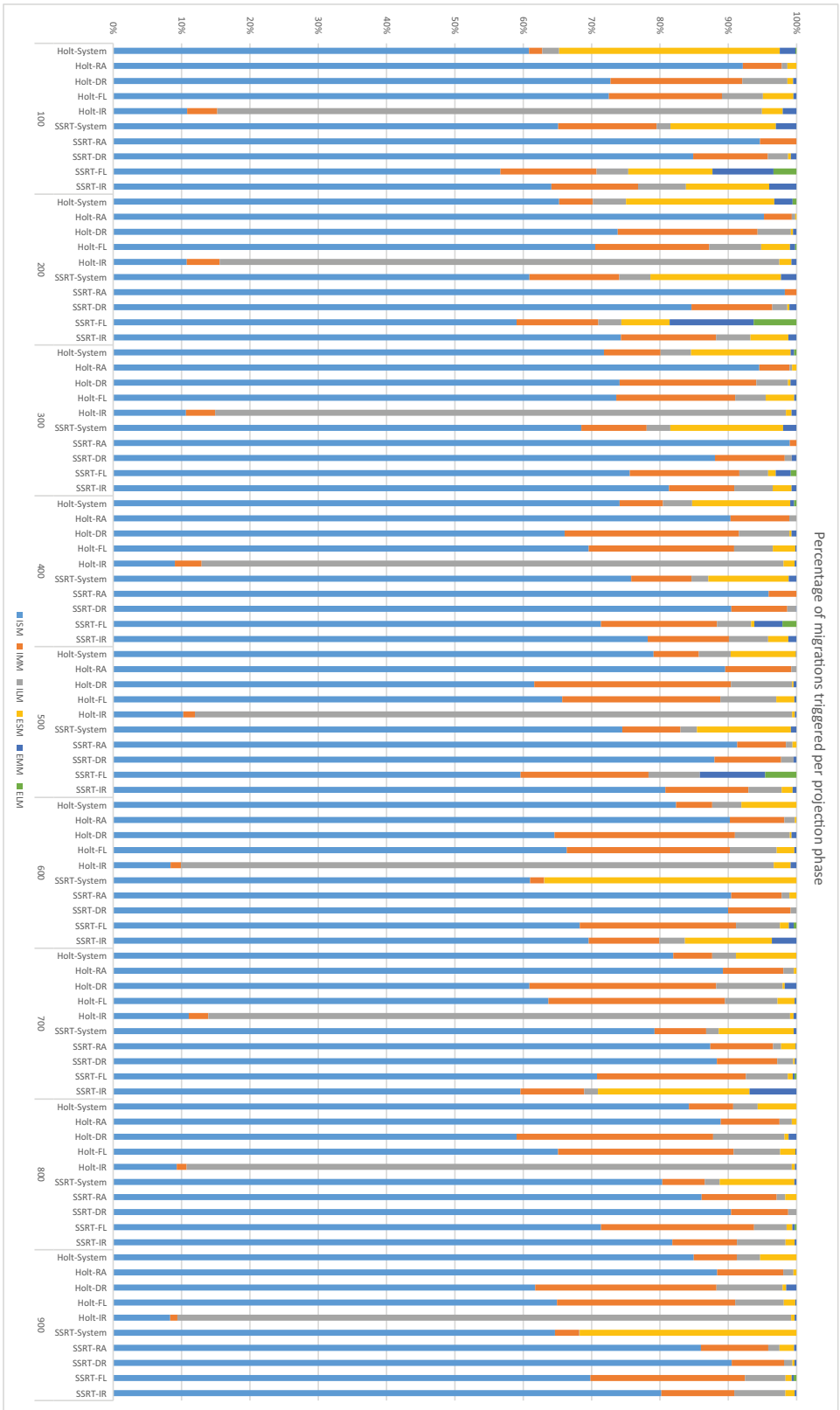


Figure 4.11: Triggered migrations per projection

projections that results in a slight improvement over Holt-FL.

In conclusion, a number of configurations have been compared on a dynamic prediction-based load balancing system for HLA-Based distributed simulations. 3 different metrics were used to evaluate the overall performance: execution time, total migrations, and the triggered migrations per projection.

Systems that implement RA show an increase in the number of the internal migrations that do not offer a benefit in reducing the execution time. DR systems, on the other hand, have the least amount of triggered migrations. This comes to an expense of high execution time. Systems that use thresholds employ restrictions to migrations within the cluster allowing for more candidates to be selected for interdomain migrations. Threshold-based systems end up in migrating loads from low capability resources to high capability ones.

As SSRT-DR provides a %24 increase in execution time and %23 decrease in number of migrations when compared to Holt-System, this makes SSRT-System a better solution than the other proposed approaches when clusters are geographically distributed and any network delay might affect the overall execution time. However, for adjacent clusters, SSRT-System performs 8.5% less migrations than original method and 7.5% less execution time as SSRT gives a better insight of the predicted load.

Chapter 5

Conclusion

The necessity of load balancing systems arises in distributed simulation environments when a better performance in terms of execution time is required. As application developers strive to make faster simulations, different systems and schemes have been proposed. Some of these systems migrate simulation entities from one shared resource to another based on the dependency of simulation entities and communication behavior. Other systems concentrate on other aspects, such as the re-arrangement of computational needs for simulation entities. Another group of systems use prediction models to identify load imbalances in shared resources. *Communication-based* load balancing systems aim to recognize simulation entities that decrease the performance of the distributed simulation. *Computational-based* load balancing systems migrate the simulation entities between shared resources, based on the computation load of the simulation entity itself or on the shared resource. *Prediction-based* load balancing systems project the load of shared resources at certain times, into the future, or more specifically, a number of future balancing cycles. The work in this thesis is based on the Prediction-based load balancing systems.

5.1 Final Remarks

In more detail, two main weaknesses are identified in the implementation of Holt's model in dynamic load balancing system for HLA-based distributed simulations: 1) the linear relationship between the projection and the time interval and 2) slow load oscillations detection. A number of Holt's model variants are proposed. Cutoff restricts unrealistic projections by applying the concept of a threshold. Reversed Trend reverses the sign of

the computed trend when its tendency does not match the actual tendency to enhance the load oscillations detection. Separate Systems introduces a separate prediction engine for each projection in order to get more precise predictions. Separate System and Reversed Trend are combined into one system, Separate System Reversed Trend, that aims to use the power of each system to provide better projections. Genetic Algorithms are used to make Holt's model adapt to any kind of load it might face. A complexity analysis was then performed on all methods to examine the overhead of each variant. Moreover, an accuracy test was established to compare the accuracy of each variant when it was implemented on dynamic load balancing for HLA-Based distributed simulations. As a result, Separate System Reversed Trend was chosen to be compared against Holt's model in real simulations.

As a second and important phase, the federate migration decision making approach was replaced with other approaches to remove the dependency of the used approach on predefined thresholds. The new approaches use different metrics to initial migration calls. Restricted Approach is a rule-based expert system that uses a set of loose rules to identify the need of triggering a migration call. Direction Based is another rule-based expert system that makes use a ratio along side the metrics used in in Restricted Approach where the ratio is used to extra restrictions on the migrations decision making. Fuzzy Logic assigns the resources to different sets in order to identify the number of migrations needed to converge the resource towards a balanced status faster. Impact Approach examines the impact of all possible migrations in a balancing cycle and gives more importance to migrations with higher impacts. These approaches were analyzed in term of time and space complexity. Moreover, we explored the behavior of all the migrations triggered by all approaches on data samples.

The last step compared the different configurations of the variants and decision making approaches. Systems that implement DR showed a lower number of triggered migrations as DR restricted the migrations by applying tough rules. The evaluations proved that SSRT variant gave a better insight of the system as it produced low error projections, and this helped in having a more efficient system.

5.2 Future Work

In this section, some of the future work is presented. These works aim to further improve the overall performance of the prediction-based dynamic load balancing system. The improvements are divided into 2 main areas: 1) Prediction Models and 2) Federate

Migration Decision-Making.

1. Prediction Model: The only prediction model used in this work was Holt's model. In the future, we will be looking into investing time implementing some other prediction models, linear or non-linear, and see what improvements we can bring to the implementation of the different model in real distributed load balancing system.

2. Dynamic Fuzzy Set Generator: as shown in FL, its drawback is that it tries to converge the resources fast to a balanced state. However, the current set generator assigns at least one resource to way overloaded set and one as way underloaded set. This results in an increase in the number of the triggered migrations. The problem with the current system is that it assigns a resource as *extremely overloaded* based on the list of load presented in the current balancing cycle. Building sets based on the current loads overuses the *extremely overloaded*. A dynamic generator is to be developed where it builds sets based on historical data rather than using on the current available list of loads. This means that the number of sets generated each balancing cycle is different and it reflects the actual status of the systems, based on historical data.

3. The proposed federate migration decision making approaches are type-independent. This means that the same approaches can be used in different types of dynamic load balancing systems for HLA-Based distributed simulations. Analyzing the effectiveness of implementing the proposed approaches in different types would be interesting.

Appendix A

Detailed Evaluations

Table A.1 shows the details of the number of migrations triggered by each configuration. The table is slit into groups, where each group shows the average error and the 95% confidence interval when a certain number of federates is executed. Moreover, it lists the ratio between each number of migration to the best case. Table A.2 shows as well the average error, ratio, and confidence for the execution time each configuration took to finish processing the simulation.

Table A.1: Detailed Number of Migrations

	Holt-Sys	Holt-RA	Holt-DR	Holt-FL	Holt-IR	SSRT-Sys	SSRT-RA	SSRT-DR	SSRT-FL	SSRT-IR
100 feds										
Avg	114.00	164.20	156.00	229.40	152.80	116.00	119.33	101.00	170.60	140.40
Ratio	1.13	1.63	1.54	2.27	1.51	1.15	1.18	1.00	1.69	1.39
Conf	16.90	11.90	16.21	24.42	13.35	17.50	16.90	24.18	17.12	3.90
200 feds										
Avg	176.60	278.40	257.00	389.00	248.80	188.80	256.75	241.00	333.00	246.00
Ratio	1.00	1.58	1.46	2.20	1.41	1.07	1.45	1.36	1.89	1.39
Conf	28.70	12.70	9.08	10.90	5.44	14.60	26.60	25.24	28.80	12.07
300 feds										
Avg	305.80	373.67	364.00	551.80	341.80	291.40	364.00	337.50	498.60	346.20
Ratio	1.05	1.28	1.25	1.89	1.17	1.00	1.25	1.16	1.71	1.19
Conf	9.60	9.40	9.08	16.50	10.50	10.90	14.30	10.30	24.00	9.60
400 feds										
Avg	422.60	450.80	435.80	695.20	413.60	403.60	458.00	381.80	643.40	422.80
Ratio	1.11	1.18	1.14	1.82	1.08	1.06	1.20	1.00	1.69	1.11
Conf	11.23	13.69	24.51	9.40	16.50	12.30	14.60	9.50	43.70	8.60
500 feds										
Avg	492.60	536.60	509.80	815.00	489.40	475.20	547.40	429.60	738.00	496.40
Ratio	1.15	1.25	1.19	1.90	1.14	1.11	1.27	1.00	1.72	1.16
Conf	30.90	20.20	9.70	17.40	6.40	18.50	33.86	27.50	45.30	9.30
600 feds										
Avg	589.00	628.60	586.40	945.40	561.40	538.60	626.80	484.00	916.20	584.33
Ratio	1.22	1.30	1.21	1.95	1.16	1.11	1.30	1.00	1.89	1.21
Conf	36.06	9.40	12.80	21.30	17.70	14.60	5.14	13.60	11.62	29.40
700 feds										
Avg	678.75	707.20	643.00	1077.80	658.40	639.60	694.00	542.60	1009.00	641.80
Ratio	1.25	1.30	1.19	1.99	1.21	1.18	1.28	1.00	1.86	1.18
Conf	38.40	13.80	23.17	26.40	18.47	23.37	10.30	25.70	79.26	11.90
800 feds										
Avg	744.80	780.60	684.80	1180.80	743.00	722.00	779.80	591.40	1094.60	727.00
Ratio	1.26	1.32	1.16	2.00	1.26	1.22	1.32	1.00	1.85	1.23
Conf	25.50	31.35	21.40	26.30	21.50	27.45	23.03	16.70	67.50	23.09
900 feds										
Avg	832.60	873.20	784.00	1338.80	844.80	794.60	859.17	636.80	1249.00	805.00
Ratio	1.31	1.37	1.23	2.10	1.33	1.25	1.35	1.00	1.96	1.26
Conf	40.90	24.70	34.08	13.50	55.16	50.16	18.50	24.50	83.18	21.80

Table A.2: Detailed Execution Time

	Holt-Sys	Holt-RA	Holt-DR	Holt-FL	Holt-IR	SSRT-Sys	SSRT-RA	SSRT-DR	SSRT-FL	SSRT-IR
100 feds										
Avg	525.92	609.20	598.80	604.20	578.80	507.60	543.00	604.00	583.00	591.40
Ratio	1.04	1.20	1.18	1.19	1.14	1.00	1.07	1.19	1.15	1.17
Conf	54.30	39.80	29.70	42.50	30.70	47.06	27.13	19.70	36.90	35.40
200 feds										
Avg	816.26	1011.20	972.20	999.00	951.60	848.00	959.50	1013.00	965.20	942.00
Ratio	1.00	1.24	1.19	1.22	1.17	1.04	1.18	1.24	1.18	1.15
Conf	19.18	33.95	31.80	45.60	43.80	65.01	58.11	55.85	69.70	54.10
300 feds										
Avg	1211.60	1362.67	1355.40	1354.80	1312.20	1162.20	1328.60	1370.00	1307.80	1333.60
Ratio	1.04	1.17	1.17	1.17	1.13	1.00	1.14	1.18	1.13	1.15
Conf	47.12	19.40	14.20	39.70	52.40	67.33	39.40	32.60	20.10	21.60
400 feds										
Avg	1511.60	1653.00	1657.80	1694.00	1576.60	1527.80	1644.80	1685.00	1701.40	1642.40
Ratio	1.00	1.09	1.10	1.12	1.04	1.01	1.09	1.11	1.13	1.09
Conf	55.07	47.15	63.25	12.90	131.90	49.02	44.30	21.20	50.12	47.70
500 feds										
Avg	1808.20	1992.80	1957.80	1989.80	1953.80	1789.00	1960.20	2003.00	1960.60	1985.80
Ratio	1.01	1.11	1.09	1.11	1.09	1.00	1.10	1.12	1.10	1.11
Conf	63.30	42.01	41.14	51.70	21.80	57.90	54.13	45.50	69.01	31.35
600 feds										
Avg	2139.40	2314.60	2298.40	2312.00	2149.40	1977.40	2305.80	2303.60	2370.49	2139.33
Ratio	1.08	1.17	1.16	1.17	1.09	1.00	1.17	1.16	1.20	1.08
Conf	108.14	34.60	34.15	46.34	123.40	194.16	28.30	31.01	54.66	283.50
700 feds										
Avg	2509.25	2626.60	2610.60	2653.40	2586.40	2483.80	2576.20	2610.80	2660.40	2339.80
Ratio	1.07	1.12	1.12	1.13	1.11	1.06	1.10	1.12	1.14	1.00
Conf	62.67	34.90	62.20	29.20	47.15	64.20	47.30	20.69	47.85	86.01
800 feds										
Avg	2791.60	2919.60	2906.80	2957.60	2913.80	2816.00	2933.40	2960.60	2994.20	2961.40
Ratio	1.00	1.05	1.04	1.06	1.04	1.01	1.05	1.06	1.07	1.06
Conf	26.70	54.97	67.50	64.31	68.90	71.20	75.10	22.42	64.70	33.50
900 feds										
Avg	3191.60	3270.20	3262.40	3346.40	3313.60	3166.20	3269.17	3286.20	3351.60	3285.60
Ratio	1.01	1.03	1.03	1.06	1.05	1.00	1.03	1.04	1.06	1.04
Conf	57.10	29.37	98.25	67.30	95.60	46.40	35.30	44.90	31.72	69.90

Bibliography

[1] Ieee standard for modeling and simulation (m amp;s) high level architecture (hla) - framework and rules. *IEEE Std. 1516-2000*, pages i –22, 2000.

[2] S. Abdullah, N. Sapii, S. Dir, and T. M. T. Jalal. Application of univariate forecasting models of tuberculosis cases in kelantan. In *Statistics in Science, Business, and Engineering (ICSSBE), 2012 International Conference on*, pages 1–7, Sept 2012.

[3] O. Abumansoor and A. Boukerche. A secure cooperative approach for nonline-of-sight location verification in vanet. *Vehicular Technology, IEEE Transactions on*, 61(1):275–285, 2012.

[4] H. Avril and C. Tropper. The dynamic load balancing of clustered time warp for logic simulation. In *Proceedings of the Tenth Workshop on Parallel and Distributed Simulation*, PADS '96, pages 20–27, Washington, DC, USA, 1996. IEEE Computer Society.

[5] A. Bakirtzis, V. Petridis, S. Kiartzis, M. Alexiadis, and A. Maissis. A neural network short term load forecasting model for the greek power system. *Power Systems, IEEE Transactions on*, 11(2):858–863, May 1996.

[6] A. Bamis, A. Boukerche, I. Chatzigiannakis, and S. Nikoletseas. A mobility aware protocol synthesis for efficient routing in ad hoc mobile networks. *Computer Networks*, 52(1):130 – 154, 2008. (1) Performance of Wireless Networks (2) Synergy of Telecommunication and Broadcasting Networks.

[7] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello. A new adaptive middleware for parallel and distributed simulation of dynamically interacting systems. In *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, pages 178–187, Oct 2004.

[8] L. Bononi, G. D'Angelo, and L. Donatiello. Hla-based adaptive distributed simulation of wireless mobile systems. In *Parallel and Distributed Simulation, 2003. (PADS 2003). Proceedings. Seventeenth Workshop on*, pages 40–49, June 2003.

[9] A. Boukerche. An adaptive partitioning algorithm for conservative parallel simulation. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 1353–1358, April 2001.

[10] A. Boukerche. *Handbook of algorithms for wireless networking and mobile computing*. CRC Press, 2005.

[11] A. Boukerche. *Algorithms and protocols for wireless, mobile Ad Hoc networks*, volume 77. John Wiley & Sons, 2008.

[12] A. Boukerche and L. Bononi. Simulation and modeling of wireless, mobile, and ad hoc networks. *Mobile ad hoc networking*, pages 373–409, 2004.

[13] A. Boukerche, I. Chatzigiannakis, and S. Nikoletseas. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. *Computer communications*, 29(4):477–489, 2006.

[14] A. Boukerche, X. Cheng, and J. Linus. Energy-aware data-centric routing in microsensor networks. In *Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 42–49. ACM, 2003.

[15] A. Boukerche, X. Cheng, and J. Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wireless Networks*, 11(5):619–635, 2005.

[16] A. Boukerche and S. K. Das. Dynamic load balancing strategies for conservative parallel simulations. *SIGSIM Simul. Dig.*, 27(1):20–28, June 1997.

[17] A. Boukerche and S. K. Das. Reducing null messages overhead through load balancing in conservative distributed simulation systems. *J. Parallel Distrib. Comput.*, 64(3):330–344, Mar. 2004.

[18] A. Boukerche, S. K. Das, and A. Fabbri. Analysis of a randomized congestion control scheme with dsdv routing in ad hoc wireless networks. *Journal of Parallel and Distributed Computing*, 61(7):967–995, 2001.

[19] A. Boukerche, S. K. Das, and A. Fabbri. Swimnet: a scalable parallel simulation testbed for wireless and mobile networks. *Wireless Networks*, 7(5):467–486, 2001.

[20] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. Sdar: a secure distributed anonymous routing protocol for wireless and mobile ad hoc networks. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 618–624, Nov 2004.

[21] A. Boukerche and X. Fei. A coverage-preserving scheme for wireless sensor network with irregular sensing range. *Ad hoc networks*, 5(8):1303–1316, 2007.

[22] A. Boukerche and X. Fei. A voronoi approach for coverage protocols in wireless sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 5190–5194. IEEE, 2007.

[23] A. Boukerche and R. E. D. Grande. Optimized federate migration for large-scale hla-based simulations. In *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*, pages 227–235, 2008.

[24] A. Boukerche and R. E. D. Grande. Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems. In *Proceedings of the 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '09, pages 175–183, Washington, DC, USA, 2009. IEEE Computer Society.

[25] A. Boukerche, S. Hong, and T. Jacob. A distributed algorithm for dynamic channel allocation. *Mobile Networks and Applications*, 7(2):115–126, 2002.

[26] A. Boukerche, S. Hong, and T. Jacob. An efficient synchronization scheme of multimedia streams in wireless and mobile systems. *Parallel and Distributed Systems, IEEE Transactions on*, 13(9):911–923, 2002.

[27] A. Boukerche and K. Lu. Optimized dynamic grid-based ddm protocol for large-scale distributed simulation systems. In *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, pages 6 pp.–, April 2005.

[28] A. Boukerche, N. J. McGraw, C. Dzermajko, and K. Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *Simulation Symposium, 2005. Proceedings. 38th Annual*, pages 259–266, April 2005.

[29] A. Boukerche and M. S. N. A. Notare. Behavior-based intrusion detection in mobile phone systems. *Journal of Parallel and Distributed Computing*, 62(9):1476–1490, 2002.

[30] A. Boukerche, H. A. B. Oliveira, E. F. Nakamura, and A. A. F. Loureiro. Localization systems for wireless sensor networks. *Wireless Communications, IEEE*, 14(6):6–12, December 2007.

[31] A. Boukerche, H. A. B. Oliveira, E. F. Nakamura, and A. A. F. Loureiro. Secure localization algorithms for wireless sensor networks. *Communications Magazine, IEEE*, 46(4):96–101, April 2008.

[32] A. Boukerche, H. A. B. Oliveira, E. F. Nakamura, and A. A. F. Loureiro. Vehicular ad hoc networks: A new challenge for localization-based systems. *Comput. Commun.*, 31(12):2838–2849, July 2008.

[33] A. Boukerche, R. W. N. Pazzi, and R. B. Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 560–567. IEEE, 2005.

[34] A. Boukerche, R. W. N. Pazzi, and J. Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Computer Communications*, 31(11):2716 – 2725, 2008. End-to-End Support over Heterogeneous Wired-Wireless Networks.

[35] A. Boukerche and Y. Ren. A secure mobile healthcare system using trust-based multicast scheme. *Selected Areas in Communications, IEEE Journal on*, 27(4):387–399, 2009.

[36] A. Boukerche and A. Roy. Dynamic grid-based approach to data distribution management. *Journal of Parallel and Distributed Computing*, 62(3):366–392, 2002.

[37] A. Boukerche and C. Tropper. A static partitioning and mapping algorithm for conservative parallel simulations. In *Proceedings of the Eighth Workshop on Parallel and Distributed Simulation*, PADS '94, pages 164–172, New York, NY, USA, 1994. ACM.

[38] A. Boukerche and M. Zhang. Towards peer-to-peer based distributed simulations on a grid infrastructure. In *Simulation Symposium, 2008. ANSS 2008. 41st Annual*, pages 212–219, April 2008.

[39] C. Burdorf and J. Marti. Load balancing strategies for time warp on multi-user workstations. *The Computer Journal*, 36(2):168–176, 1993.

[40] W. Cai, S. J. Turner, and H. Zhao. A load management system for running hla-based distributed simulations over the grid. In *Distributed Simulation and Real-Time Applications, 2002. Proceedings. Sixth IEEE International Workshop on*, pages 7–14, 2002.

[41] C. Carothers and R. M. Fujimoto. Background execution of time warp programs. In *In Proceedings of the 10th Workshop on Parallel and Distributed Simulation (PADS*, pages 12–19, 1996.

[42] C. D. Carothers and R. M. Fujimoto. Efficient execution of time warp programs on heterogeneous, now platforms. *Parallel and Distributed Systems, IEEE Transactions on*, 11(3):299–317, Mar 2000.

[43] K. M. Chandy and J. Misra. Distributed simulation: A case study in design and verification of distributed programs. *Software Engineering, IEEE Transactions on*, SE-5(5):440–452, Sept 1979.

[44] W. Charytoniuk, M. S. Chen, and P. Van Olinda. Nonparametric regression based short-term load forecasting. *Power Systems, IEEE Transactions on*, 13(3):725–730, Aug 1998.

[45] M. Choe and C. Tropper. On learning algorithms and balancing loads in time warp. In *Parallel and Distributed Simulation, 1999. Proceedings. Thirteenth Workshop on*, pages 101–108, 1999.

[46] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA, 2000.

[47] R. E. D. Grande and A. Boukerche. Distributed dynamic balancing of communication load for large-scale hla-based simulations. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 1109–1114, June.

[48] R. E. D. Grande and A. Boukerche. Dynamic load redistribution based on migration latency analysis for distributed virtual simulations. In *Haptic Audio Visual Environments and Games (HAVE), 2011 IEEE International Workshop on*, pages 88–93, Oct.

[49] R. E. D. Grande and A. Boukerche. Predictive dynamic load balancing for large-scale hla-based simulations. In *Distributed Simulation and Real Time Applications (DS-RT), 2011 IEEE/ACM 15th International Symposium on*, pages 4–11, Sept.

[50] E. Deelman and B. K. Szymanski. Dynamic load balancing in parallel discrete event simulation for spatially explicit problems. In *Parallel and Distributed Simulation, 1998. PADS 98. Proceedings. Twelfth Workshop on*, pages 46–53, May 1998.

[51] E. El Ajaltouni, A. Boukerche, and M. Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *Distributed Simulation and Real-Time Applications, 2008. DS-RT 2008. 12th IEEE/ACM International Symposium on*, pages 61–68, Oct 2008.

[52] M. Elhadef, A. Boukerche, and H. Elkadiki. A distributed fault identification protocol for wireless and mobile ad hoc networks. *Journal of Parallel and Distributed Computing*, 68(3):321–335, 2008.

[53] R. F. Engle, C. Mustafa, and J. Rice. Modelling peak electricity demand. *Journal of Forecasting*, 11(3):241–251, 1992.

[54] E. A. Feinberg and D. Genethliou. Chapter 12 load forecasting.

[55] J. Feng. A new method for ionospheric short-term forecast using similar-day modeling. In *Antennas, Propagation EM Theory (ISAPE), 2012 10th International Symposium on*, pages 472–474, Oct 2012.

[56] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, Jun 2002.

[57] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, Aug. 2001.

[58] R. M. Fujimoto. Parallel discrete event simulation. *Commun. ACM*, 33(10):30–53, Oct. 1990.

[59] 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 *Parallel and Distributed Simulation, 2000. PADS 2000. Proceedings. Fourteenth Workshop on*, pages 139–146, 2000.

[60] E. S. Gardner. Exponential smoothing: The state of the art - part ii. *International Journal of Forecasting*, 22(4):637 – 666, 2006.

[61] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[62] D. W. Glazer and C. Tropper. On process migration and load balancing in time warp. *Parallel and Distributed Systems, IEEE Transactions on*, 4(3):318–327, Mar 1993.

[63] R. E. D. Grande and A. Boukerche. Dynamic balancing of communication and computation load for hla-based simulations on large-scale distributed systems. *J. Parallel Distrib. Comput.*, 71(1):40–52, Jan. 2011.

[64] T. Haida and S. Muto. Regression based peak load forecasting using a transformation technique. *Power Systems, IEEE Transactions on*, 9(4):1788–1794, Nov 1994.

[65] K. L. Ho, Y. Y. Hsu, C. F. Chen, T. E. Lee, C. C. Liang, T. S. Lai, and K. K. Chen. Short term load forecasting of taiwan power system using a knowledge-based expert system. *Power Systems, IEEE Transactions on*, 5(4):1214–1221, Nov 1990.

[66] O. Hyde and P. F. Hodnett. An adaptable automated procedure for short-term electricity load forecasting. *Power Systems, IEEE Transactions on*, 12(1):84–94, Feb 1997.

[67] D. R. Jefferson. Virtual time. *ACM Trans. Program. Lang. Syst.*, 7(3):404–425, July 1985.

[68] M. Jiang, R. Anane, and G. Theodoropoulos. Load balancing in distributed simulations on the grid. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3232–3238 vol.4, Oct 2004.

[69] M. R. Jiang, S. P. Shieh, and C. L. Liu. Dynamic load balancing in parallel simulation using time warp mechanism. In *Parallel and Distributed Systems, 1994. International Conference on*, pages 222–227, Dec 1994.

[70] S. J. Kiartzis, A. G. Bakirtzis, J. B. Theocharis, and G. Tsagas. A fuzzy expert system for peak load forecasting. application to the greek power system. In *Electrotechnical Conference, 2000. MELECON 2000. 10th Mediterranean*, volume 3, pages 1097–1100 vol.3, May 2000.

[71] Z. Li, W. Cai, S. J. Turner, and K. Pan. Federate migration in a service oriented hla rti. In *Distributed Simulation and Real-Time Applications, 2007. DS-RT 2007. 11th IEEE International Symposium*, pages 113–121, 2007.

[72] K. Liu, S. Subbarayan, R. R. Shoults, M. T. Manry, C. Kwan, F. L. Lewis, and J. Naccarino. Comparison of very short-term load forecasting techniques. *Power Systems, IEEE Transactions on*, 11(2):877–882, May 1996.

[73] M. Y. H. Low. Dynamic load-balancing for bsp time warp. In *Simulation Symposium, 2002. Proceedings. 35th Annual*, pages 267–274, April 2002.

[74] J. Luthi and S. Grossmann. The resource sharing system: dynamic federate mapping for hla-based distributed simulation. In *Parallel and Distributed Simulation, 2001. Proceedings. 15th Workship on*, pages 91–98, 2001.

[75] V. Miranda and C. Monteiro. Fuzzy inference in spatial load forecasting. In *Power Engineering Society Winter Meeting, 2000. IEEE*, volume 2, pages 1063–1068 vol.2, 2000.

[76] J. Misra. Distributed discrete-event simulation. *ACM Comput. Surv.*, 18(1):39–65, Mar. 1986.

[77] M. Mohandes. Support vector machines for short-term electrical load forecasting. *International Journal of Energy Research*, 26(4):335–345, 2002.

[78] U. of Chicago. Globus, 2008.

[79] H. A. B. Oliveira, A. Boukerche, E. Freire Nakamura, and A. A. F. Loureiro. An efficient directed localization recursion protocol for wireless sensor networks. *Computers, IEEE Transactions on*, 58(5):677–691, May 2009.

[80] H. A. B. Oliveira, A. Boukerche, E. F. Nakamura, and A. A. F. Loureiro. Localization in time and space for wireless sensor networks: An efficient and lightweight algorithm. *Perform. Eval.*, 66(3-5):209–222, Mar. 2009.

[81] P. Peschlow, T. Honecker, and P. Martini. A flexible dynamic partitioning algorithm for optimistic distributed simulation. In *Principles of Advanced and Distributed Simulation, 2007. PADS '07. 21st International Workshop on*, pages 219–228, June 2007.

[82] C. S. R. Schlagenhaft, M. Ruhwandl and H. Bauer. Dynamic load balancing of a multi-cluster simulator on a network of workstations. *SIGSIM Simul. Dig.*, 25(1):175–180, July 1995.

[83] Y. Ren and A. Boukerche. Modeling and managing the trust for wireless and mobile ad hoc networks. In *Communications, 2008. ICC'08. IEEE International Conference on*, pages 2129–2133. IEEE, 2008.

[84] S. Ruzic, A. Vuckovic, and N. Nikolic. Weather sensitive method for short term load forecasting in electric power utility of serbia. *Power Systems, IEEE Transactions on*, 18(4):1581–1586, Nov 2003.

[85] S. Samarah, M. AlHajri, and A. Boukerche. A predictive energy-efficient technique to support object-tracking sensor networks. *Vehicular Technology, IEEE Transactions on*, 60(2):656–663, 2011.

[86] J. S. Steinman, C. A. Lee, L. F. Wilson, and D. M. Nicol. Global virtual time and distributed synchronization. *SIGSIM Simul. Dig.*, 25(1):139–148, July 1995.

[87] G. Tan and K. C. Lim. Load distribution services in hla. In *Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004. Eighth IEEE International Symposium on*, pages 133–141, Oct 2004.

[88] J. W. Taylor, L. M. de Menezes, and P. E. McSharry. A comparison of univariate methods for forecasting electricity demand up to a day ahead. *International Journal of Forecasting*, 22(1):1 – 16, 2006.

[89] J. W. Taylor and P. E. McSharry. Short-term load forecasting methods: An evaluation based on european data. *Power Systems, IEEE Transactions on*, 22(4):2213–2219, Nov 2007.

[90] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[91] L. Wilson and W. Shen. Experiments in load migration and dynamic load balancing in speedes. In *Simulation Conference Proceedings, 1998. Winter*, volume 1, pages 483–490 vol.1, Dec 1998.

[92] L. Wu, J. Yan, and Y. Fan. Data mining algorithms and statistical analysis for sales data forecast. In *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*, pages 577–581, June 2012.

[93] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary. Scheduling critical channels in conservative parallel discrete event simulation. In *Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation*, PADS '99, pages 20–28, Washington, DC, USA, 1999. IEEE Computer Society.

[94] K. Zajac, M. Bubak, M. Malawski, and P. Sloot. Towards a grid management system for hla-based interactive simulations. In *Distributed Simulation and Real-Time Applications, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 4–11, Oct 2003.