Distributed Dynamic Balancing of Communication Load for Large-Scale HLA-based Simulations †

Robson Eduardo De Grande
School of Information Technology and Engineering
University of Ottawa - Ottawa, Canada
Email: rdgrande@site.uottawa.ca

Azzedine Boukerche
School of Information Technology and Engineering
University of Ottawa - Ottawa, Canada
Email: boukerch@site.uottawa.ca

***Abstract*—In large-scale distributed simulations, communication aspects are highly significant due to their direct influence on performance. The High Level Architecture (HLA) provides services for managing such simulations and reducing their communication overhead. However, HLA does not present any solution for the communication latencies caused by the network distances among simulation elements. Several dynamic balancing schemes have been proposed attempting to provide a general best solution for the performance issues caused by computation and communication imbalances. Amongst these schemes, some just perform a limited redistribution of communication load. Based on a proximity analysis of federate interactions, a distributed dynamic scheme for balancing the communication load of HLA-based simulations is devised. The design of this distributed scheme aims at improving fault tolerance, decreasing communication and computation overload, and avoiding bottlenecks in the system. The distributed balancing system, organized in the hierarchical structure, monitors simulations, redistributes load, and migrates federates. Experiments have been realized to compare the proposed distributed scheme with a centralized scheme and to prove its effectiveness for large-scale HLA-based simulations.**

***Index Terms*—Distributed Systems; High Level Architecture; Communication Latency**

I. Introduction

Large-scale distributed simulations highly depends on communication, so such simulations are susceptible to overhead and networking delays. The High Level Architecture (HLA) provides mechanisms to organize distributed simulations and diminish the transfer of irrelevant data. However, even with such tools, HLA-based simulations can undergo interaction latencies caused by communication distances. Communication latencies affect simulation performance according dynamic, unpredictable changes of federates' interactions. A static partitioning cannot determine such dynamic changes, so a dynamic scheme for balancing communication load is needed to adapt the simulation distribution properly. The dynamic scheme also needs to present a distributed structure to minimize balancing overhead, improve reactivity, and provide fault tolerance.

The HLA framework [1] was designed to delimit a standard for designing and coordinating distributed simulations. These standard consists of a set of rules, interface specifications, and object model templates to enable re-usability and interoperability of simulation elements, called federates. Consequently, a distributed simulation is basically composed of Run Time Infrastructure (RTI) services and federates that act according to the HLA specification. In the RTI, the Data Distribution Management (DDM) restricts the data transfers that are relevant for federates in a simulation. Nevertheless, even though this approach reduces the consumption of network resources, it cannot minimize communication delays originated from network distances.

Many balancing schemes have been proposed to provide performance improvement for distributed simulations through redistribution of load. Most of the proposed balancing systems aim to re-configure simulation components based on computation load, which consistently improves performance. Some proposed balancing systems consider communication factors in their schemes. Such schemes use a communication aware decision-making computation load re-distribution, or they present a limited communication-oriented re-allocation of simulation entities. A scheme was proposed in [2] in order to expand the scope of the limited communication balancing schemes. Even though this scheme presents a hierarchical structure, it might be susceptible to failures and cause some overload depending on the scale of system.

Therefore, a distributed scheme for balancing communication is proposed in order to improve the adaptation and reaction to dynamic communication load changes. The balancing scheme employs a proximity analysis, attempting to reduce or minimize the network latencies in HLA-based simulations. In order to re-allocate federates dynamically, the scheme, similarly to [2], is composed of three phases: monitoring, redistribution, and migration. Moreover, the proposed balancing system is organized in a hierarchical structure in which the balancing components act independently in order to allow tolerance to failures, improve scalability characteristics, and decrease balancing overhead.

The remainder of this paper is organized as follows. In section 2, the related work is presented, describing the challenging issues of dynamically balancing the communication of large-scale HLA-based simulations. In section 3, the distributed balancing system is introduced by delineating its architecture, hierarchical organization, and functioning. In the section 4, experiments are defined and their results are discussed. Finally, in section 5, the conclusion presents a brief paper outline and describes the directions for future work.

† This is work is partially supported by NSERC, the Canada Research Chair program, ORF funds, and EAR Research Award

II. Related Work

The design of load balancing systems presents high complexity due to the existence of many factors that hamper the creation of a general best solution; consequently, several balancing techniques have been proposed. Basically, all the proposed approaches consider computation load and simulation inter-dependencies in their balancing schemes. The balancing of computation load influences positively the execution time of distributed simulations, but in large-scale environments, communication delays substantially affect the simulation performance and need to be considered in by balancing schemes. Thus, there exist many approaches that observe simulation look-ahead or communication dependencies between simulation entities to improve simulation performance.

In order to decrease delays in simulation execution, some balancing schemes monitor and analyzes the simulation look-ahead or the communication rate of each simulation entity. The look-ahead enables the discovery of entities that are slowing down a simulation [3], [4]. The analysis of communication inter-dependencies is used to determine the critical interaction flow; these dependencies are analyzed statically [5] [6] or dynamically [7], [4], [8], [9], [10], [11], [12], [13]. All these techniques redistribute load to eliminate network latencies by migrating simulation entities to the same resource to which they communicate. This is a limited approach, so another technique was proposed in [2] to broaden the redistribution scope through proximity analyses.

The proximity analysis is more appropriate for distributed schemes since it increases the possibilities of reducing communication latencies. In [2], the proposed balancing system is organized hierarchically with a centralized data analysis. The accumulation of information can cause computation overhead in the balancing system, and the existence of only one balancing component makes it susceptible to failures. Therefore, a decentralized approach emerges as vital for a balancing scheme, allowing the existence of distributed balancing components that act independently.

III. Distributed Balancing Scheme

In the proposed distributed balancing scheme, the load managers perform their part of load balancing independently from the others. As a consequence, the synchronization existent among the load management elements is avoided, minimizing delays and preventing failures. To detect imbalances, the recent past is used as the predictive tool in the balancing scheme. However, not the most up-to-date status information of a distributed system is employed because not the whole system information is collected at the same time. For balancing, the schemes employs an architecture similar to the one described in [2]. Additionally, modifications in the inter-relations between the scheme elements are introduced, so a distributed redistribution algorithm can be applied. In the approach therefore, data transfers occur only when re-arrangement of load is necessary, decreasing the flow of collected load data.

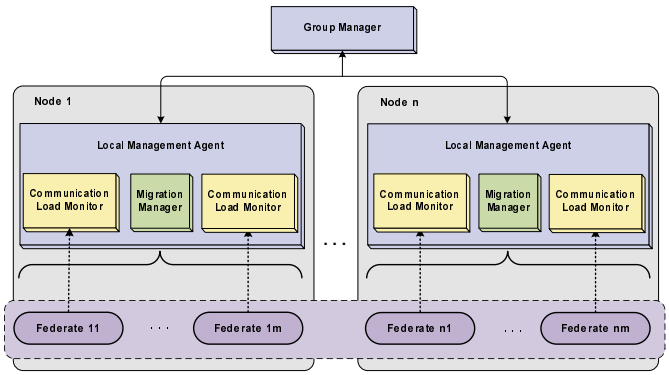


Fig. 1: The Dynamic Communication Balancing's General Architecture

A. Architecture and Balancing Phases

As depicted in fig 1, the Group Manager (GM) is the main balancing element in the system architecture. A GM is responsible for managing a set of federates and resources, determining a domain in the distributed environment. The GM controls all the balancing steps: monitoring, redistribution, and migration. The component requests information from the resources and federates through Local Management Agents, analyzes the collected data, re-allocates federates, and emits migration calls. The Local Management Agent (LMA) acts as an interface to the federates. The LMA provides CPU utilization and communication behaviour of each federate, and it forwards migration calls from GMs to federates. In LMA, the Communication Load Monitor is responsible for collecting information, and the Migration Manager (MM) is the component that forwards and manages migration calls through Grid services. The GM also accesses Grids services in order to obtain computational load of the shared resources.

In order to constantly obtain up-to-date data about resources load status, a resource management system becomes essential. Grid computing was proposed as a resource sharing system involving the coordination of resources, individuals, and institutions [14]. Moreover, Globus Toolkit [15], based on Open Grid Services Architecture (OGSA) [16], is the de facto implementation for a Grid computing middleware. The Grid services range the monitoring of resources and applications, the scheduling and allocation of resources, and the identification of application requirements [17]. Such Grid services are accessed by the balancing system in order to retrieve CPU load information and transfer federates.

The balancing algorithm is basically organized in monitoring, redistribution, and migration phases. In the monitoring phase, an intra-domain data collection is performed by a GM and each LMA. The collected data regards computation load of each resource, computation load produced by each federate, and communication behaviour of each federate. After data is collected, a GM realizes data filtering, detects communication imbalances, and selects federate candidates for migration. In the redistribution phase, analyses are performed through comparisons between the communication rates of federate

candidates and a local overall average. This re-allocation of federates is performed between GM peers, considering that the remote GM in the pair is the one responsible for determining the re-partitioning. When the migration moves are determined, they are returned to the GM that originated the redistribution call. In the migration phase, a GM uses the migration call information from remote GMs to identify the respective federate candidate, send the call to it, and initiate the two-phase migration process. In the first part of migration, Grid services are accessed in order to submit a federate remotely and perform the data transfer of its static state information. In the second part of migration, a MM employs a migration proxy in the migration procedure. The proxy allows a peer-to-peer transfer of a federate's execution state. This migration process minimizes the migration latency and improves reactivity and performance.

B. Distributed Re-partitioning Algorithm

In a centralized redistribution of load, all involved parts are synchronized in order to keep the consistency of actions and provide correct information, introducing global synchronization in the balancing system. Such a synchronization generates delays and overheads and makes the balancing system susceptible to failures. More specifically, the balancing that requires a global view of a distributed system is impaired by this synchronization approach. Since the communication balancing needs to be aware of entire system, it is based on a centralized approach and might experience such synchronization issues. Consequently, if the load management element, GM, that is root in the hierarchical architecture fails, the sub management elements are influenced and cannot provide a sub-balancing, leading to the failure of the entire balancing system.

The introduction of asynchronous relations between hierarchical, independent GMs avoids the issues caused by the centralization of re-distribution procedure. As a result, GMs report only to their neighbours in the hierarchy to emit calls in order to re-allocate federates. The receiver of such calls analyzes the them in order to accept federates for re-allocation, and it concludes process in a redistribution move. The analysis consists of asynchronous comparisons between the communication rate of the received federates with its local federates, observing the availability of resources.

The proposed distributed re-partitioning phase attempts to act as it has the global view of the entire distributed environment. In a HLA-based simulation, all federates produce different communication loads and they are placed on all the available resources, presenting different distances for their communication. Consequently, the GMs of the balancing system are organized in layers, respectively according to their distances. The organization respects the order of distances, so each GM is aware of its posterior and anterior GMs in the hierarchical structure. For this alignment, each Group Manager keeps two thresholds in order to organize the simulation federates that it is managing. The two thresholds represent the maximum and minimum communication rates based on the sample data gathered during a balancing cycle. Regarding the amount of the communication that federates performed in the recent past, these thresholds identify federates that do not match with the communication distance of the resources on where they are running. These bounds are obtained through the observation of communication rate and the relations between Group Managers.

Therefore, α and β are established for each Group Manager (GM). As a result of the system's hierarchical organization, it is assumed that $\alpha_{l1} > ... > \alpha_{ln}$ and $\beta_{l1} > ... > \beta_{ln}$ are sequences organized in descending order, observing that all $\alpha_{ln} > \beta_{ln}$ and all $\beta_{ln} \geq \alpha_{ln-1}$. The coherency among the bounds αs and βs is induced by the communication load of resources and is kept through the inter-communication between GMs, originated by federate migration calls from neighbour GMs. The adjustments of these bounds occur during runtime at each balancing cycle when new sample data is gathered. Additionally, a feedback related to previous migration move calls is employed to re-calculate the bounds, so they are calibrated according to the success in achieving migrations.

1) Migration Moves: As a result of distributing the re-partitioning algorithm and employing bounds to organized the balancing system, federate candidates for migration are not sent directly to the most proper destination. Because of the distributed structure of the balancing system, the federates are passed in a peer-to-peer fashion for re-distribution. Consequently, based on top and bottom bounds of a GM and according to communication rates, a federate is transferred gradually until it reaches the most proper position. Depending on the architectural hierarchy of the balancing system, the process of moving a federate to its final destination may take several balancing cycles. The process produces delays in the redistribution and overhead by performing all the required migrations. In order to avoid these time, communication, and computation expenses, a jump condition is introduced.

2) Group Manager Inter-Relations: The distributed balancing system creates the sense that tall the hierarchical system acts as centralized to provide a pseudo global environment view. To produce such a view, rules are required to regulate the bounds of each GM, and migration move policies are introduced. Consequently, the GMs need to have some topological knowledge about the environment, so they can act independently but respecting migration move policies. The balancing elements are placed according to the topological structure of the distributed environment, and knowledge about the generated balancing structure is explicitly passed to every GM. Such knowledge consists of the balancing elements that are placed before and after a GM according to both their communication distances.

Based on the environment topology, each Group Manager manages its inner bounds, α and β, to move federates according to the communication needs and the topological organization of the distributed system. In such a hierarchical organization (root, intermediate, and leaf GMs), the root's α contains value infinite and the leafs' β contains value -1 or minus infinite, delimiting the limiting boundaries of the balancing system: the closest position and the furthest position

in every branch of the structure. The organization and the relations between GMs produces the following situations for calibrating the balancing bounds:

$$\beta_1 = \alpha_2, \beta_2 = \alpha_3, ..., \beta_{n-1} = \alpha_n. \quad (1)$$

$$\beta_{n-1} < alpha_n or \beta_{n-1} \ll alpha_n. \quad (2)$$

$$\beta_{n-1} > alpha_n or \beta_{n-1} \gg alpha_n. \quad (3)$$

The expression 1 denotes the ideal communication distribution state of all bounds, representing a completely balanced system. However, such a situation is not achieved due to dynamic changes that occur during run-time.

The expression 2 represents the overlapping of two sequential domains. In this case, this approach minimizes the number of redistribution calls, and imbalances are more hardly detected, decreasing the balancing reactivity.

The expression 3 generates a discontinuous breach between GMs' domains. The breach causes the constant generation of redistribution calls. Such calls are needed in the structure in order to regulate GMs' thresholds constantly and adapt to communication changes promptly. Through the inter-relation policies, the calls force each GM directly or indirectly regulate the variables of its interacting GMs. In a direct regulation, a call is sent to a sub GM to be considered in the re-calculation of its α. In a indirect regulation, the non-acceptance of redistribution calls incites the sender GMs to adapt their α values accordingly. Therefore, the balancing follows the condition 3 as a rule to adjust bounds of acceptance ranges. Following this rule, three situations determine the redistribution behaviour of a Group Manager: the presence of computation underload, the existence of computation overload, and the arrival of a federate candidate with extreme communication rate.

The first situation expresses the availability of resources for receiving load. This situation occurs when the number of federates is smaller than the number of federates that can be supported by the available resources managed by a Group Manager. A GM in such a situation needs to enforce the move of federates from its sub GMs to its resources. As a result, a GM lowers its β (bottom bound) and incites its sub GMs to decrease their α (upper bound). With this tactic, the GM allows more federate candidates to its resources.

On the other hand, the second situation represents the lack of availability of resources. If the resources are not able to admit any more federates, federate moves and exchanges are realized to re-organize the load and decrease the communication overload. In this case, a GM narrows its acceptance range, inducing federates to be migrated to resources managed by other GMs and relieving the load of local resources. Moreover, federate exchanges enable the re-allocation of federates to benefit the computational load and communication balance.

The third situation describes the case in which the communication load of federates in redistribution calls exceeds or falls behind both thresholds of a GM. The abrupt extreme changes in communication rates of federates cause the balancing system to require several balancing cycles to completely transfer such a highly communicative federate to its final correct position, which is not the next GM in the hierarchy. Therefore, to speed up the transferring process, a GM allows a federate to jump several distances (GMs) until the federate reaches the proper resource for its communication rate. Thus, when the communication rate of a federate in a redistribution call exceeds the opposite bound of a GM, the GM just forwards it to the next known GM in the hierarchy. If the communication rate is high enough, this procedure is repeated many times, and the forwarding GMs await an answer from the destination GM.

3) Threshold Adjustment: The adjustments of bounds occur asynchronously and periodically according the load rearrangement needs. Focusing on an inter-relation between a GM and its up GM, it is observed that for the GM's α and its up GM's β the balancing system attempts to reach the relation $\alpha \simeq \beta$ and $\alpha \leq \beta$. Because each Group Manager works independently, the balancing analysis of federates is based on only collected data samples. Moreover, due to balance only communication, the balancing system is federate-centred and focus on federate communication metrics. Consequently, the use of standard deviation leads to flexibility, so the bounds adapt accordingly when communication rate increases or decreases. However, due to the existence of inter-relations between GMs, each GM needs to correct or adapt its bounds to its upper and lower GMs, so a merge between asynchronous adjustments and self-redistribution are employed in the system,

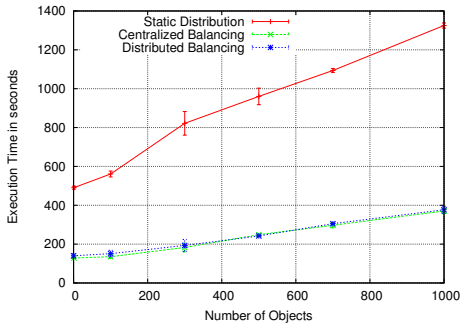
$$\alpha_{ln-1} = \alpha_{ln-1} * (1 - \theta) + \beta_{ln} * (\theta), \theta \geq 0. \quad (4)$$

The parameter θ is used as a control to assure the dependency relations between GMs. In the scheme, θ is related to the amount of success in generating migration moves from the federate candidates that are received: $moves/candidates$.

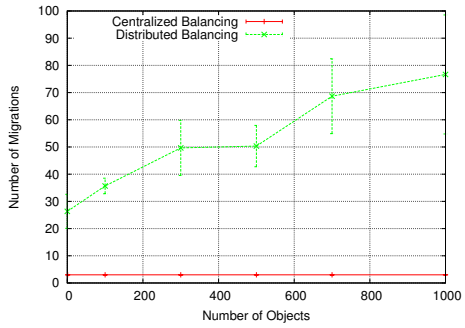
IV. Experimental Results

Even though the distributed scheme increments the qualities of the balancing system, it needs to react properly to communication imbalances. Therefore, three experimental analyses have been accomplished, which embraced the execution of large-scale HLA-based simulations on two computing clusters connected by a fast-Ethernet link. One of the clusters consisted of a set of 24 computing servers, which contained a Quadicore 2.40GHz Intel CPU and 8 GB of RAM memory, and it had a 2-Gb Myrinet optical network connection. The other cluster comprised of 32 computing servers, which were constituted of a Core 2 Duo 3.4 GHz Intel CPU and 2 GB of RAM, and it used a Gb Ethernet network connection. In this environment, Linux operating system and the Globus Toolkit 4.2.1 were installed to support all the experiments, and the HLA platform with RTI version 1.3 was used to coordinate the simulations.

For the experiments, the simulation elements were placed evenly on an environment of 55 computing nodes. The HLA RTI executive that coordinated the simulations was placed on a dedicated server. In the same way, the proposed balancing system was deployed on each computing server of the environment. Each server received a Local Management Agent and both management nodes of the clusters ran Group Managers.

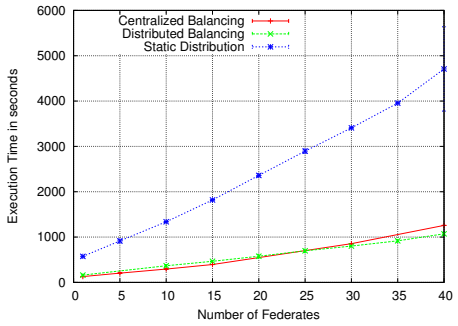


(a) Performance Comparison

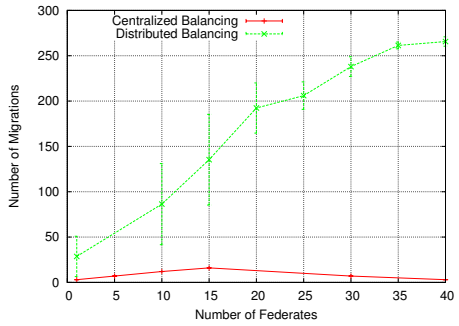


(b) Comparison of Number of Migrations

Fig. 2: Communication Balancing Gain for an Increasing Communication Load



(a) Performance Comparison



(b) Comparison of Number of Migrations

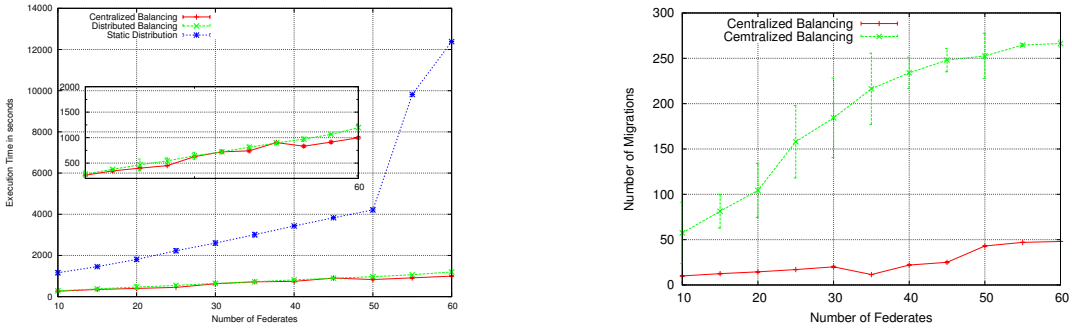
Fig. 3: Communication Balancing with an Increasing Number of Communicative Federates

The simulation scenario consisted in training operations coordinated in a two-dimension routing space. The federates organized the movement two teams of interactive tanks in time-stepped simulations. Each federate was responsible for calculating the movement of a tank, publishing a tank's position, and subscribing to other tank's updates. The simulations were composed of 500 federates that managed 1 to 1000 objects in 100 time steps. To insert controlled communication latency imbalances, some federates published data related to special objects, generating large communication overhead.

The reaction to a restricted communication imbalance is observed in the first experiment. An increasing communication overhead caused by a small number of federates was introduced in the simulations. The overhead includes the publication of updates by a federate and the subscription to them by other three federates. As shown in fig 2, the execution time of the simulations is directly influenced by the increase of communication load, mainly for the baseline. The centralized and distributed balancing systems presented a substantial performance gain when compared with the baseline curve in fig 2a. However, in fig 2b, there is a noticeable difference in the number of migrations. The centralized approach only performed the migration moves to improve the communication latencies for the communicative federates, but the distributed approach continuously realized migrations while there were available resources with smaller communication distance.

The reactivity of the proposed balancing system to increasing communication overhead is analyzed in the second experiment. The analysis evaluates the balancing response to a high communication overload, which consists in inducing an overhead produced by 100 special objects in 1 to 40 federates. In this case, the balancing system needs to perform several migration moves with the objective of minimizing the communication latencies. As depicted in fig 3, the baseline presented high augment in simulation execution time as the number of communicative federates increased. As shown in the graph in fig 2a, the centralized and distributed approaches presented a similar performance gain in fig 3a. The graph in fig 3b evidences that the distributed approach realized a larger number of migrations to achieve the same decrease of simulation time than the centralized did. Moreover, as observed in the graph, the centralized approach did not move all the communicative federates in the simulation, denoting its limitation in detecting some overloads.

The efficiency in detecting and reacting to dynamic changes of communication load is evaluated in the third experiment. Similarly to the previous experiments, the simulations were composed of 1 to 60 communicative federates that performed data updates. The updates of these communicative federates contained a random number of special objects that ranged between 1 to 100, inducing a dynamic and unpredictable behaviour. As depicted in fig 4a, the balanced simulations outperformed the baseline simulation, but more specifically after the number of 50 federates, the baseline presented a significant increase in the simulation time caused by the cumulative saturation of communication. When comparing



(a) Performance Comparison

(b) Comparison of Number of Migrations

Fig. 4: Communication Balancing with an Increasing Number of Communicative Federates

only the balancing systems, they both showed similar performance gain, but with differences in the number of migrations, as shown in fig 4b. The centralized balancing realized less migrations than the distributed did, showing that the distributed approach has a strong tendency to react to slight changes of communication load.

Therefore, the distributed balancing scheme showed similar performance to a centralized balancing scheme. Even though the distributed approach presented larger number of migrations, the simulation performance was not compromised. This results from migration latencies being minimized by the performance gain provided by redistributions. As consequence, the distributed balancing could achieve a reasonable simulation efficiency improvement and add characteristics that avoid bottlenecks, overheads, and failures.

V. Conclusion

A distributed dynamic balancing scheme is proposed to overcome the drawbacks of a centralized scheme for large-scale HLA-based simulations. The proposed scheme aims to avoid a single point of failure, communication and computation overhead, and system bottlenecks. The proposed scheme is composed of monitoring, redistribution, and migration, and it is organized hierarchically. In the hierarchical structure, the balancing components act independently and asynchronously. Moreover, Grid services are used to provide monitoring data and reliable federate migration.

The experiments proved that the distributed balancing system is as effective as the centralized one. However, the distributed approach presented a larger number of migrations than the centralized approach did. This increase reflects the higher reactivity of the distributed system, but the system might produce precipitated modifications in some specific cases. Thus, as future work, further experimental analysis will be realized to adjust the balancing system's parameters and decrease the number of migration moves. Moreover, larger-scale experiments will be performed to more precisely evaluate the benefits of the distributed balancing scheme.

References

[1] S. I. S. C. (SISC), "Ieee standard for modeling and simulation (m&s) high level architecture (hla) framework and rules," IEEE Computer Society, September 2000.

[2] R. E. D. Grande and A. Boukerche, "Dynamic partitioning of distributed virtual simulations for reducing communication load," in *Proc of the IEEE Int Workshop on Haptic Audio visual Environments and Games*. IEEE Computer Society, To appear. 2009.

[3] B. P. Gan, Y. H. Low, S. Jain, S. J. Turner, W. C. W. J. Hsu, and S. Y. Huang, "Load balancing for conservative simulation on shared memory multiprocessor systems," in *Proc. of the workshop on Parallel and distributed simulation*. IEEE Computer Society, 2000, pp. 139–146.

[4] M. Y. H. Low, "Dynamic load-balancing for bsp time warp," in *Proc. of the Annual Simulation Symposium*. IEEE Computer Society, 2002, pp. 267–274.

[5] R. Schlagenhaft, M. Ruhwandl, and C. S. H. Bauer, "Dynamic load balancing of a multi-cluster simulator on a network of workstations," in *Proc. of the workshop on Parallel and distributed simulation*. IEEE Computer Society, 1995, pp. 175–180.

[6] M. Choe and C. Tropper, "On learning algorithms and balancing loads in time warp," in *Proc, of the workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 101–108.

[7] J. Jiang, R. Anane, and G. Theodoropoulos, "Load balancing in distributed simulations on the grid," in *Proc. of the Int. Conference on Systems, Man and Cybernetics*. IEEE Computer Society, 2004, pp. 3232–3238.

[8] P. Peschlow, H. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," in *Proc. of the Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 2007, pp. 219–228.

[9] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary, "Scheduling critical channels in conservative parallel discrete event simulation," in *Proc. of the workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 20–28.

[10] A. Boukerche and C. Tropper, "A static partitioning and mapping algorithm for conservative parallel simulations," in *Proc. of the workshop on Parallel and distributed simulation*. IEEE Computer Society, 1994, pp. 164–172.

[11] A. Boukerche, "An adaptive partitioning algorithm for conservative parallel simulation," in *Proc. of the Int. Parallel and Distributed Processing Symposium*. IEEE Computer Society, 2001, pp. 133–138.

[12] E. E. Ajaltouni, A. Boukerche, and M. Zhang, "An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure," in *Proc. of the Int. Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 61–68.

[13] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello, "An adaptive load balancing middleware for distributed simulation," in *Workshop on Middleware and Performance (WOMP)*, 2006, pp. 864–872.

[14] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200–222, 2001.

[15] "Globus," University of Chicago, 7 Feb. 2008. [Online]. Available: http://www.globus.org/

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

[17] J. Nabrzyski, J. M. Schopf, and J. Weglarz, *Grid Resource Management: State of the Art and Future Trends*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.