Dynamic Partitioning of Distributed Virtual Simulations for Reducing Communication Load †

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

***Abstract*—HLA-based simulations can experience performance degradation due to communication latencies between simulation federates, which generate significant cumulative overhead. Even though the HLA standard provides mechanisms to decrease the misuse of network resources, it does not present any tool to diminish the communication latencies between interactive federates. Moreover, the interaction dependencies can be predicted before simulations are initiated, but such predictions relies on the determinism of simulations, producing erroneous balancing when simulations change their load dynamically. Thus, an hierarchical three-phase dynamic communication load balancing scheme is devised to react to run-time load changes, so the scheme performs constant, periodical monitoring of resources, re-distribution of load, and migration of federates. The balancing system reorganizes the distribution of large-scale HLA-based simulations, so the communication latencies are minimized, increasing the parallelism of the distributed simulations and leading to a performance improvement. Experiments were realized to measure the benefits of the scheme, and through comparative analyses, the balancing scheme presented considerable performance improvement to HLA-based simulations.**

I. Introduction

Large-scale HLA-based distributed simulations can undergo load imbalances that are generate by irregular computation or communication load partitioning or by dynamic run-time load changes. Mostly dependent on the configuration of HLA simulations, the intra-dependencies of simulations cause communication imbalances due to the network latencies and distances between interactive federates. A static initial partitioning can avoid these communication imbalances for determinist simulations in which all the simulation actions are known beforehand, but in the presence of unpredictable run-time load changes, the same partitioning hardly reaches an efficient deployment of simulation entities. As a result, in order to maximize the performance of large-scale distributed simulations, a dynamic communication load balancing scheme is proposed.

The High Level Architecture (HLA) standard [1] was designed to manage parallel and distribute simulations. HLA-based simulations are composed of independent, interactive federates that are organized by the HLA specification and administrated by the Run Time Infrastructure (RTI) management services. Furthermore, the RTI provides services to minimize the communication latencies in HLA simulations through the Data Distribution Management (DDM) service. This service restricts the message transmission to relevant communication between interacting federates, minimizing the consumption of network resources by the overall simulation with unnecessary messages. However, the standard does not provide any mechanism to decrease the communication latencies between highly interactive federates. Thus, even with DDM, an HLA simulation can experience communication overhead due to the network distances between federates.

In order to improve the performance of distributed simulations, several dynamic load balancing schemes were proposed in the literature. Such approaches mostly attempt to managed the computation load of simulations, but some of them consider communication latency as significant factor in balancing the load of simulations. However, the existent solutions that consider communication just re-allocate the load of distributed simulations by moving a simulation entity to the resource that the entity communicates the most. Even though this technique succeeds in managing peer-to-peer simulations, it is limited and restricts the re-distribution of load considerably.

In this paper, a dynamic communication load balancing scheme is proposed to minimize the communication overhead in large-scale HLA simulations. In such a scheme, the re-partitioning of load considers the proximity of resources of a distributed system. Moreover, the scheme is designed in an hierarchical architecture to avoid balancing overhead caused by the monitoring and to facilitate the management of the resources. In order to detect and react to dynamic load changes, the proposed balancing system constantly performs monitoring of federates, re-distribution of load, and migration of federates.

The remainder of the paper is organized as follows. In section 2, a review of the related work and the challenging issues are presented. In section 3, the proposed hierarchical three-phase scheme for balancing communication load is introduced. In section 4, experiments and their results described and discussed. In section 5, the conclusion and directions for future work are outlined.

II. Related Work

There exist many balancing approaches in the literature that attempt to re-distribute the load of simulations dynamically in order to provide a performance improvement. Such approaches offer solutions to optimistic and conservative sim-

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

ulations and re-deploy simulation entities according to the computation and communication load imbalances. Regarding the communication load, the existent solutions manage the load by considering simulation look-aheads or the interaction dependencies between simulation entities. Nevertheless, such solutions realize a static partitioning or are limited.

Computation load and simulation speed are considered by many balancing techniques. Some of these techniques retrieve the CPU utilization of simulation entities or the throughput of simulations to determine load imbalances [2] [3] [4] [5] [6] [7] [8]. Other techniques employ the processing speed of simulation entities as a metric to evaluate the balance of simulations [9] [10] [11] [12]. However, all of these approaches cannot detect the overhead in the simulation time that is produced by communication latencies.

In order to minimize the delays generated by the dependencies between simulation entities, look-ahead and communication load are observed in distributed simulations. Some approaches present simulation-dependent solutions that analyse and limit the look-ahead of simulations to decrease the delays between simulation entities [13] [14]. Other approaches analyse the communication dependencies statically [15] [16] or dynamically [17] [14] [18] [19] [20] [21] [22] [23]. Nevertheless, all these solutions evidence simulation-dependent techniques, limit the parallelism of simulations, or do not consider the communication topology of shared resources.

The existent solutions for communication load balancing are limited and cannot be applied fully to HLA-based distributed simulations. The simulation-independent solutions perform re-distribution of load disregarding the topology of the shared resources. Even though this approach produces some performance improvement through the elimination of communication latencies, it is limited to peer-to-peer simulations and does not observe the proximity of resources when moving load. Besides the creation of more possibilities to decrease latencies, the neighborhood analysis of resources allows the balancing of communication load for HLA simulations based in centralized RTIs. As a result, a dynamic load balancing system that employs communication topology analysis in its decision-making mechanism is proposed to minimize the communication overhead of HLA-based simulations.

III. Dynamic Communication Load Balancing

The proposed balancing scheme aims to provide a simulation-independent, dynamic communication load re-partitioning mechanism for large-scale HLA-based distributed simulations. Such a scheme is organized in the three phases that periodically detect and react to the dynamic interaction changes in the managed simulations. Furthermore, in order to minimize the balancing overhead generated by the constant monitoring of resources, re-distribution of load, and migration of federates, an hierarchical architecture and a low-latency federate migration technique are employed in the scheme's design. According to the design depicted in the figure 1, the balancing scheme is basically composed of a Group Manager and several Local Management Agents, which consist of a Migration Manager and a Communication Load Monitor. Aiming at a cross-platform design, all these components are implemented in Java.

The Group Manager (GM) administrates all the balancing scheme, organizing the required balancing tasks and performing the re-distribution algorithms. The manager controls the monitoring, re-distribution, and migration mechanisms. The monitoring mechanism gathers the collected information from each Local Management Agent (LMA) and from each sub-GM in the hierarchical deployment of the balancing system. After the data gathering is performed, the monitoring mechanism filters the information and searches for load dissimilarities in the data sample, so communication imbalances can be identified, triggering the re-distribution mechanism. The re-distribution mechanism re-partitions the federates according to their communication needs, which are detected by the monitoring mechanism. After federates are re-allocated to the proper resources, the migration calls are forwarded to the respective Local Management Agent.

A Local Management Agent (LMA) is placed in each managed resource in order to gather the monitoring information from each federate and forward migration calls. The monitoring information is gathered through a list of Communication Load Monitors, which are components that periodically collect the communication rate of federates. Moreover, a LMA is responsible for forwarding migration calls to Migration Managers. In order to avoid simulation causality inconsistencies, the Migration Manager supports all the steps of federate migration. The Manager realizes the migration procedure by launching the Migration Manager at a remote resource, suspending and restoring a federate's execution, and managing the exchange of data required to restore a migrating federate properly. The Migration Manager also triggers a Migration Proxy to assist in the data transfers required by the migration processes. When the destination resource of a migration call is not reachable directly by the source resource to perform peer-to-peer transmission of data, the proxy is employed to act as an intermediate element, establishing temporary connections with both communicating parts to forward migration data.

In order to decrease the amount of overhead that the balancing generates in the distributed system and to facilitated the management of distributed simulations, a multi-layered configuration is employed in the design. In such hierarchical architecture, each GM is responsible for managing a set of LMAs and a set of GMs. Each LMA corresponds to a shared resource in the system, and it corresponds to the end-point in the architecture. The bottom GMs in the hierarchy controls just LMAs and are managed by other GMs, the GM father. In the middle of the hierarchy, the a GM additionally controls a list of other GMs and is managed by a father GM. At the top of the hierarchy, there is a GM that controls all the sub-GMs directly or indirectly. The top GM comprehends the endpoint of the hierarchy by gathering of monitoring information, performing the re-distribution of load, and originating all the migration moves.

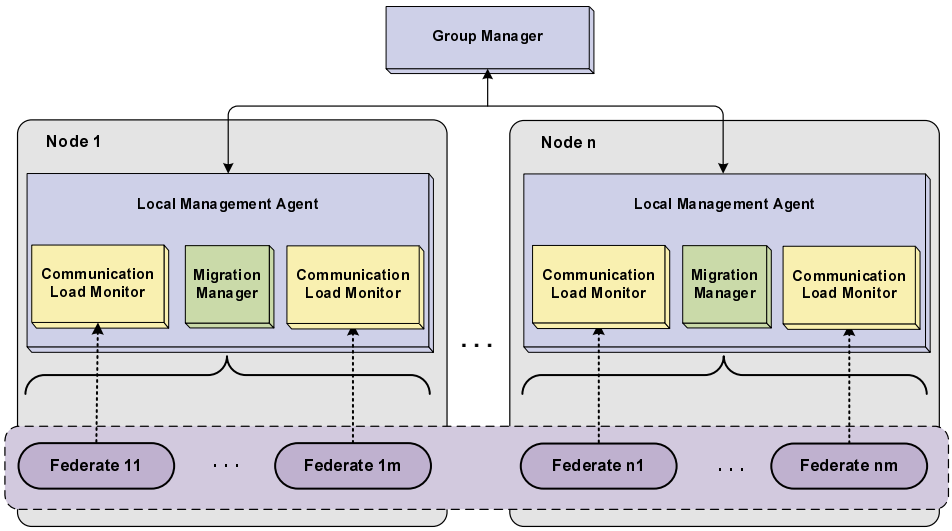

Fig. 1. The Dynamic Communication Balancing's General Architecture

A. Monitoring Phase

The monitoring phase is the initial phase in the reactive dynamic communication balancing scheme and triggers the subsequent phases to re-partition and move load. The phase identifies the need for federate re-distribution through environmental analyses that evidences communicative federates in simulations. In order to determine communicative federates, the interactions of each federate are kept in the communication table. Such a table locates locally in each resource and logs every the federate interaction by recording the destination address, the number of messages sent and received by the federate, and their size. Since the balanced simulations are implemented in a HLA RTI centralized architecture, the monitored federates communicate directly with the same destination resource, the RTI, so the federates are classified only by their communication rate. This registered information is collected by the Local Communication Monitor every monitoring interval to perform filtering and selection.

Selection is employed to identify determinant aspects of decision factors in the collected data sample. These determinant aspects regards discrepancies in the communication rates of federates and evidence the need for re-allocation of resources to decrease the communication latencies. The selection consist in to compare all the the federate's communication rates with an average in order to differentiate the federates that interact the most in a managed simulation. To obtain such an average, an arithmetic mean is calculated, and a threshold is used to delimit more accurately the communicative federates with comparisons. The threshold is retrieved by the calculation of the standard deviation of the analyzed data sample and is employed as the superior boundary, which evidences the communication imbalances caused by interactive federates. The standard deviation is used in the data sample analysis since the communication rate is an application-dependent metric and presents a behavior that changes according to simulation implementations. As a result, the interactive federates are selected and are analysed in the next balancing phase in order to produce re-organizations of load to decrease the communication overhead.

B. Re-distribution Phase

The re-partition of communication load searches the the most appropriate resources for the highly communicative federates after the need for re-allocation of resources is identified in the monitoring phase. In order to identify the destination resources for re-partitioning of federates, the algorithm classifies the shared resources according to their communication topology, determining the distances of the resources to a specific target, the RTI. Therefore, basically, the re-distribution moves the communicative candidates to the resources that are closer to the RTI, expecting to decrease the overhead caused by the communication latencies.

After the ordered list of communicative federates is determined, each of these federates is evaluate according to the re-distribution algorithm. Based on the location where a federate is, the algorithm seeks for a destination resource. In order to perform such a search, a structure called Path Distances is employed to determine the closest location to the target resource that the federate can be transferred, so the communication latency can be decreased. The optimal destination for a federate transfer is the resource that it communicates the most since the communication latencies are minimized or eliminated. As with it was shown in the previous works, simulation entities with excessive common communication are grouped in the same resource in order to eliminate the networking delays and consequently exclude the communication latencies. Nevertheless, this approach causes the overload of a resources and decreases the simulation parallelism, so it cannot be applied to HLA-based simulations based on centralized RTIs. As a result, federates are not migrated to the resource where the RTI is running, but they

are migrated to the closest resources to the RTI. The path distances are employed to determine such resources through a set of distance rings, and each ring contains a set of destination candidates that can receive the communicative federates.

The Path Distances is a data structure that stores information about the communication topology and is employed by the balancing system to search for destination resources to perform communication balancing. Analyzing the communication topology, this data structure classify the resources in distance rings. Resources that have the same distance are grouped in the same ring in the path distances, so a search for a destination resource is easily realized when employing such a structure. The distance corresponds to the sum of the number of hops between two resources, including the networking latency of each hop, and the resources in each ring are classified according to their computation load in order for the balancing system to select a resource with harming the distribution of computation load. As a matter of simplification, this structure is created just once in the start up of the balancing system since the distance the balanced application is an HLA simulation based on a centralized RTI.

A resource candidate is determined by selecting the resource with least load from the distance ring closest to the RTI. After such a resource is identified, computation load comparisons are realized to analyse the migration move, so it does not overload the candidate resource, considering the average load of the rest of the shared resources. The candidate resource's computation load is compared with the overall computation load of the resources, and the difference between them needs to obey the policies delimited for computation load distribution [24] in order not to harm the computation load balance. If the resource candidate cannot receive the communicative federate, the next candidate is searched in distance rings. The next ring in the classification order is selected, and its resource with the least load is selected. The same comparisons are performed to determine a migration move for the communicative federate. The process of selecting the distance rings continues while the destination resource is not found or the communication latency of the ring is smaller than the latency of the resource where the federate is. When the algorithm reaches a distance ring with a communication latency equal to or larger than the federate's resource, the re-distribution algorithm stops since it cannot improve the communication latency of that federate regarding the load configuration that distributed system presents in that moment.

C. Migration Phase

In the migration phase, all the federate migration moves generated in the re-distribution phase are issued to their respective Migration Managers. Because federate migration can introduce a considerable overhead to a simulation's execution time, the migration latency is directly related to the reactivity of the balancing system, so minimal migration latency allows a better performance improvement of the balanced simulation. The federate migration procedure corresponds to transfer the initialization files of a federate to the destination resource, suspend the federate, retrieve its execution state and the incoming messages, transfer the state and the messages to the destination, and restore the federate execution. The data transfers performed in the migration procedure add significant overhead to the simulation, which increases even more when performing migration in large-scale distributed systems. As a result, in order to minimize the migration latency, a two-phase federate migration technique is employed in the balancing scheme. Similarly to the migration techniques proposed by Boukerche and De Grande [25] and Zengxiang et. al. [26], the technique is freeze-free, requires minimal external tools, and avoids unnecessary communication and computing to be accomplished.

In the first phase of the migration technique, all the static initialization information is transmitted to the remote resource, so the federate and Migration Manager can be configured and be ready to perform the next steps in the migration procedure. These initialization information is transmitted through GridFTP [27], and the migrating federate and the Migration Manager are initiated in the remote through the Web Service Grid Resource Allocation and Management (WS GRAM) [27]. These third-party tools produce a considerable overhead, but such overhead is not incorporated to the simulation execution time because the federate does not suspend its execution while the data transfers are performed.

After the initialization files are successful transferred and the migration is initiated in the remote resource, the second migration phase is realized. First, the Migration Manager exchanges the communication channels with the RTI in order for the migrating federate to wait for the execution state information. After the exchange is realized, the Migration Manager suspends the federate in the local resource and request its execution state to be transmitted to the Migration Manager at the remote resource. This execution state corresponds to dynamic information that represents the current execution state of a federate and its Local Run-time Controller's state. Together with the state, the messages that were received during the migration procedure and were not processed by the federate are transmitted to the remote resource. When received, the state and messages are set properly, so the migrating federate restores its execution and starts processing the messages in the correct order.

IV. Experiments and Results

Experiments have been accomplished to measure the efficiency of the proposed dynamic communication balancing system. Through the experiments, the system proved to detect and react to communication dynamic changes accordingly, decreasing the communication latencies and consequently improving the simulation performance. In order to support the experiments, a large-scale distributed environment composed of two clusters was employed. The environment consisted of a Dell cluster consisting of 24 nodes, an IBM cluster consisting of 32 nodes, and a fast-Ethernet link connecting them. In the Dell cluster, each node comprised a Quadicore 2.40GHz Intel(R) Xeon(R) CPU and 8 gigabytes of DIMM DDR RAM

memory, and all nodes were inter-connected through a Myrinet optical network that allowed data transmission up to 2 gigabits per second. In the IBM cluster, each node consisted of a Core 2 Duo 3.4 GHz Intel(R) Xeon(R) CPU and 2 gigabytes of DIMM DDR RAM, and a gigabit Ethernet network connected the cluster's nodes. In all the experimental environment, Linux operating system have been installed, and the experiments have been supported by the HLA platform with an RTI version 1.3 and the Globus Toolkit 4.2.1.

In the experimental simulations, the simulation federates were distributed evenly on the 55 nodes, and the HLA RTI executive was placed in a dedicated node. The Local Management Agents were deployed on the clusters except the nodes that were designated to the RTI and the Group Managers. A Group Manager was placed in one node of each cluster, and one Group Manager was the father. Furthermore, in the simulation scenario, federates simulated the emergency preparedness scenario of a fire fighting situation. In the simulation, federates coordinated the actions to update the information of objects that represented fire fighters, fire focuses, and buildings, in a two-dimension routing space in time-stepped simulations. Such federates updated the information of their objects that were published and subscribed to interest spaces. The simulations were composed of 500 federates that managed 1 to 1000 objects in 100 time steps. In order to produce controlled, differentiated communication latencies in the simulations, certain federates performed the publication of special objects that generates a large communication overhead.

As shown in the graph of the figure 2, the experiment observed the performance improvement of the dynamic balancing system in managing the communication load of a simulation with increasing communication. One of the 500 federates presented an increasing number of objects generating communication overhead, which reflected to the simulation execution time because of the simulation intra-dependencies. In the graph, a noticeable performance gain appears when the communicative federate contains more than 100 federates, meaning that the update of objects started to increase the communication latency considerably, and the migration of the federate produced a significant gain.

Another experiment observed the reactivity of the proposed balancing scheme to the increase of the simulation communication overhead, as depicted in the graph 3. In the experiment, 100 special objects were assigned to 1 to 100 federates. The curves in the graph proved that the dynamic re-distribution of federates detected the need for modifying the configuration of federates to decrease the communication load and achieve a simulation performance improvement.

According to the graph in the figure 4, the proposed balancing system was able to detect the dynamic changes of communication load in the simulations and react properly. In this experiment, similarly to the parameters in the previous one, 1 to 110 federates of the simulations produced a random amount of objects updates that ranged from 1 to 100. Even though this number of updates changed periodically during the simulation time, the dynamic balancing scheme successfully identified the federates with intensive communication.

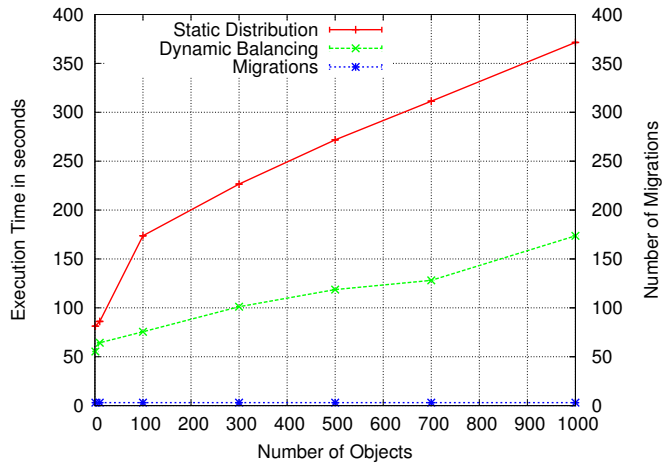

Fig. 2. The Communication Balancing Gain for an Increasing Communication Load

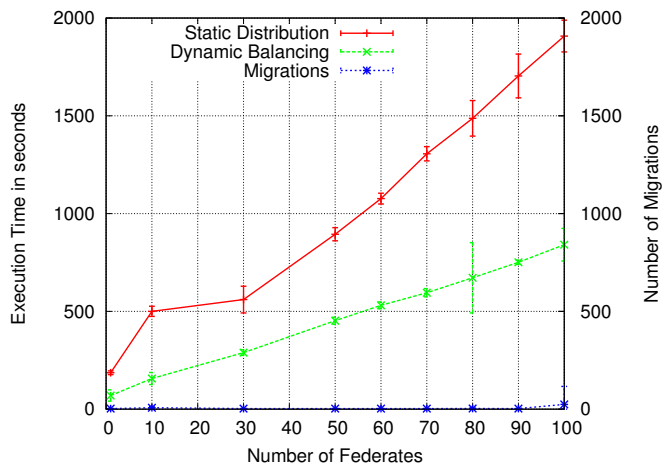

Fig. 3. Communication Balancing with an Increasing Number of Communicative Federates

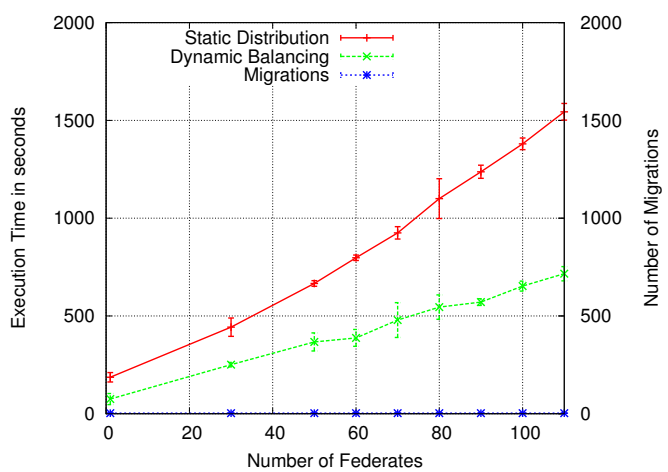

Fig. 4. Communication Balancing with Dynamic Communication Load

V. Conclusion

In this paper, a dynamic communication load balancing scheme was proposed to decrease the communication latencies of large-scale HLA-based distributed simulations. The scheme is designed in a multi-layered hierarchical architecture and is organized in three sequential phases, which performs the detection of communication imbalances, re-distribution of load, and migration of federates. Grid services are also employed in balancing system to provide computation monitoring information to aid the decision making in the re-distribution phase and to realize reliable data transfers. The experiments proved the effectiveness of the balancing system in to decrease the simulation execution times as a consequence of diminishing the communication latencies of highly interactive federates through federate migration.

As future work, aiming to achieve better performance improvement, other detection techniques and re-distribution algorithms will be employed in the balancing system. Moreover, as an extension, additional re-distribution mechanisms will be added to the scheme in order to support the communication balancing of peer-to-peer simulations, allowing the migration of simulation entities to the resource that they interact the most and performing a grouping procedure to identify imbalances and determine load re-configurations.

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] L. F. Wilson and W. Shen, "Experiments in load migration and dynamic load balancing in speedes," in *Proc. of the 1998 Winter Simulation Conference*. IEEE Computer Society, 1998, pp. 483–490.

[3] E. Deelman and B. K. Szymanski, "Dynamic load balancing in parallel discrete event simulation for spatially explicit problems," in *Proc. of the 12th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1998, pp. 46–53.

[4] A. Boukerche and S. K. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *Proc. of the 11th Workshop on Parallel and Distributed Simulation (PADS97)*. IEEE Computer Society, 1997, pp. 32–37.

[5] J. Luthi and S. Grossmann, "The resource sharing system: dynamic federate mapping for hla-baseddistributed simulation," in *Proc. of the 15th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 2001, pp. 91–98.

[6] W. Cai, S. J. Turner, and H. Zhao, "A load management system for running hla-based distributed simulations over the grid," in *Proc. of the 6th International Workshop on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2002, pp. 7–14.

[7] K. Zajac, M. Bubak, M. Malawski, and P. Sloot, "Towards a grid management system for hla-based interactive simulations," in *Proc. of the 7th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Comp. Society, 2003, pp. 4–11.

[8] H. Avril and C. Tropper, "The dynamic load balancing of clustered time warp for logic simulation," in *Proc. of the 10th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 1996, pp. 20–27.

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

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

[11] C. D. Carothers and R. M. Fujimoto, "Background execution of time warp programs," in *Proc. of the 10th Workshop on Parallel and Distributed Simulation*. IEEE Computer Society, 1996, pp. 12–19.

[12] G. Tan and K. C. Lim, "Load distribution services in hla," in *Proc. of 8th IEEE Distributed Simulation and Real-time Applications*. IEEE Computer Society, 2004, pp. 133–141.

[13] 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 14th workshop on Parallel and distributed simulation*. IEEE Computer Society, 2000, pp. 139–146.

[14] M. Y. H. Low, "Dynamic load-balancing for bsp time warp," in *Proc. of the 35th Annual Simulation Symposium (SS02)*. IEEE Computer Society, 2002, pp. 267–274.

[15] 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 9th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1995, pp. 175–180.

[16] M. Choe and C. Tropper, "On learning algorithms and balancing loads in time warp," in *Proc, of the 13th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 101–108.

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

[18] P. Peschlow, H. Honecker, and P. Martini, "A flexible dynamic partitioning algorithm for optimistic distributed simulation," in *Proc. of the 21st Workshop on Parallel and Distributed Simulation (PADS07)*. IEEE Computer Society, 2007, pp. 219–228.

[19] Z. Xiao, B. Unger, R. Simmonds, and J. Cleary, "Scheduling critical channels in conservative parallel discrete event simulation," in *Proc. of the 13th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1999, pp. 20–28.

[20] A. Boukerche and C. Tropper, "A static partitioning and mapping algorithm for conservative parallel simulations," in *Proc. of the 8th workshop on Parallel and distributed simulation*. IEEE Computer Society, 1994, pp. 164–172.

[21] A. Boukerche, "An adaptive partitioning algorithm for conservative parallel simulation," in *Proc. of the 15th International Parallel and Distributed Processing Symposium*. IEEE Computer Society, 2001, pp. 133–138.

[22] 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 12th 2008 International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 61–68.

[23] 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.

[24] A. Boukerche and R. E. D. Grande, "Dynamic load balancing using grid services for hla-based simulations on large-scale distributed systems," in *Proc. of the 13th International Symposium on Distributed Simulation and Real-Time Applications (to appear)*. IEEE Computer Society, 2009.

[25] ——, "Optimized federate migration for large-scale hla-based simulations," in *Proc. of the 12th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2008, pp. 227–235.

[26] Z. Li, W. Cai, S. J. Turner, and K. Pan, "Federate migration in a service oriented hla rti," in *Proc. of the 11th International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, 2007, pp. 113–121.

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