Measuring Communication Delay for Dynamic Balancing Strategies of Distributed Virtual Simulations

Robson E. De Grande, Azzedine Boukerche and Hussam Ramadan

***Abstract*—As an inherent characteristic of any distributed system, the execution and performance of distributed virtual simulations totally rely on underlay communication infrastructure and resources. The performance of such simulations is directly restricted by the communication latencies between inter-dependent simulation components. The High Level Architecture (HLA) is a framework designed with the objective of organizing these simulations through management services. However, the framework is unaware of the communication delays caused by the network distances between communicating simulation parts. These delays can result from non-planned initial deployment or dynamic simulation changes, requiring constant load balancing. Due to the importance of balancing distributed simulations, many approaches have been designed. In order to provide a balancing system aware of the dynamic communication changes, a delay-based redistribution scheme has been designed. The scheme successfully arranges the load, but it lacks precision due to communication delay oscillations. Therefore, extensions are proposed to modify the balancing algorithm in order to avoid unnecessary, mistaken load rearrangements. In the experimental results, the delay-based scheme has been able to reduce the simulation execution time when compared with the distributed balancing scheme, and the proposed extension has been capable of increasing the precision of the balancing.**

***Index Terms*—Distributed Simulations, HLA, Performance, Load Analysis**

I. Introduction

The performance of distributed virtual simulations substantially relies on the communication dependencies between simulation entities that are deployed on large-scale environments. These dependencies dictate the progress of coordinated processing, and they are inevitably influenced by an underlay communication infrastructure. Communication latencies cumulatively add delays in a simulation, and most of the delays are caused by network distances and unpredictable dynamic communication load changes. These delays exhibit a stronger influence on performance as the scale of the simulation and the environment grows. The High Level Architecture (HLA) framework unfortunately does not present any approach for minimizing communication latencies on distributed virtual simulations. Thus, a balancing scheme has been designed to constantly measure the communication conditions, analyze the collected information, and promptly react to run-time changes.

The HLA framework [1] was conceived to facilitate the coordination and management of distributed simulations. The HLA framework is fundamentally composed of a set of rules, interface specifications, and object model templates, which are defined to provide interoperability and enable re-usability of simulation elements. According to the HLA specification, a simulation is called a federation and consists of one or more simulation entities, which are called federates. In order to introduce run-time simulation coordination, a Run Time Infrastructure (RTI) is also presented in the designed framework to enable the management services. More specifically, the Data Distribution Management (DDM) service not only organizes the exchange of information inside simulations but also restricts data transfers among federates. Even though this restriction technique reduces the amount of data transmitted, it is unaware of the communication latencies. As a result, delays are added to the simulation time for each simulation event exchanged.

Due to the high relevance of performance for distributed simulations, many balancing schemes have been designed. These schemes observe the computational and communication characteristics of a distributed system. Although the analysis of computational load substantially improves performance, this balancing cannot avoid the interaction waiting time between simulation entities. Distributed simulations are strongly influenced by communication due to dependencies between simulation components. As a result, the proposed balancing of HLA-based virtual simulations focus on decreasing communication delays is focused. Based on this objective, redistribution schemes have been designed in [2] [3] and [4]. In such schemes, an analysis of the proximity of resources according to the network topology is performed in their redistribution algorithms to rearrange federates. However, these schemes completely rely on the static characteristics network resources.

A redistribution algorithm centred on communication delay is proposed to overcome the issues originating from the static load balancing features of the two previous schemes. The proposed solution enables the detection of dynamic communication load changes. The proposed balancing scheme is structured in the monitoring, redistribution, and migration phases. The redistribution phase of the proposed scheme employs the proximity analysis approach

R. E. De Grande is with the School of Information and Technology Engineering, University of Ottawa, Ottawa. E-mail: rdgrande@site.uottawa.ca

A. Boukerche is with the School of Information and Technology Engineering, University of Ottawa, Ottawa, and a visiting Professor at King Saud University, Riyadh, Saudi Arabia. E-mail: boukerch@site.uottawa.ca

H. Ramadan is with the College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. E-mail: hussam@ksu.edu.sa

to increase the redistribution possibilities. The balancing system is organized in a hierarchical structure, and the balancing elements interact through relations that dynamically adapt according to the communication delays and load. Due to the volatility of communication delay, extensions are developed to increase the precision of balancing response to communication load oscillations.

The remainder of this paper is organized as follows. In Section 2, the related work and challenging issues are presented. In Section 3, the proposed balancing scheme is presented by defining the systems architecture and functioning. In Section 4, extensions for the proposed scheme are detailed. In Section 5, the experimental scenario is delineated, and results are discussed. Finally, in Section 6, a brief conclusion is presented determining directions for future work.

II. Related Work

Optimal load distribution is not a simple task and requires complex analysis of monitored data to determine run-time. Due to the importance and complexity of load balancing, many balancing schemes have been designed for optimistic and conservative distributed simulations. Essentially, these schemes attempt to observe computational or communication load in order to determine a sub-optimal re-allocation of resources or simulation entities. Measuring and analyzing computational load can considerably improve simulation performance, but all this effort might be lost if a large amount of delay in simulations originates from communication latencies. Consequently, measuring communication status and analyzing internal interactions of distributed simulations present high relevance for increasing execution performance.

The balancing schemes focused on computing resources employ a resource-centred or simulation-centred approach. According to the resource-centred technique, the balancing scheme requires measuring the status of the resources on which the simulation load is deployed [5] [6] [7] [8] [9] [10]. In summary, the redistribution algorithm attempts to maximize overall utilization of resources. Following the simulation-centred technique, the balancing scheme measures the local simulation speed of each simulation entity [11] [12] [13] [14] [15]. Simulation entities with lower speed indicate an overloaded resource, and their rearrangement benefits the simulation performance. Nevertheless, this balancing approach cannot identify delays caused by communication latencies and may not be able to determine imbalances that result from simulation dependencies.

To avoid or minimize the delay effects in simulations, look-ahead and communication rate are the most commonly employed metrics in balancing systems. Look-ahead is used by balancing schemes to indirectly reveal the dependencies that increase simulation execution time [16] [17]. The analysis of communication rates and latencies provide the means to directly determine which dependency might be causing overhead. The dependency causes delays fundamentally because of network distances and overhead. The analysis of these dependencies and reorganization of load is performed either statically through critical path analysis [18] [19] or dynamically through periodic measurement and analysis of simulation interactions [20], [17], [21], [22], [23], [24], [25], [26]. The previously designed balancing schemes are simulation-dependent, limit the parallelism of simulation processing, or disregard the network topology. To provide a solution that considers the proximity of resources, two schemes have been developed [2] [3] [4].

Based on the analysis of resources locality, a hierarchical scheme with centralized data analysis has been designed in [2] [3]. A distributed redistribution algorithm [4] is proposed to overcome the negative features of the centralized algorithm. However, both schemes are unable to detect stress of the resources caused by communication overhead of external processes or caused by saturation of network resources. A delay-based balancing scheme [27] has been developed to observe the measurement of communication delay in the load redistribution. Even though this scheme enables the adaption to more dynamic characteristics of distributed simulations and resources, it lacks precision due to the volatility of communication delay. Thus, an extension is proposed to improve the analysis of gathered communication delay measurements and consequently react to communication imbalances more properly.

III. Balancing System Based on Communication Delay

To react to dynamic communication load changes, the proposed balancing scheme incorporates periodic measurements of network conditions and communication behaviour, load analysis, and rearrangement of HLA-based simulation federates. In the balancing scheme, the most recent status past is adopted as the main prediction metric. A decentralized load redistribution algorithm is employed to prevent issues present in centralized schemes, such as single point of failure, synchronization, and overhead [4]. The algorithm acts according to the hierarchical structure that organizes the balancing components. This structure basically represents the network topology that determines the positioning of the computing resources and classifies simulations in different domains (clusters). A domain consists in a region of the environment that is monitored by a set of balancing components.

A. Architecture

Similarly to the architecture in [4] and as depicted in Figure 1, the proposed load balancing system relies on Group Managers (GM) to coordinate all the balancing steps. This component manages the resources and simulation federates inside a domain, and it accesses information from other domains to detect imbalances. The balancing steps that are organized by a GM are comprised of monitoring, redistribution, and migration. Monitoring is the initial part in each balancing cycle and requires the measurement of simulation and resource aspects. The balancing metrics are obtained through a Monitoring Interface component and Local Management Agents (LMA). Based on such metrics,

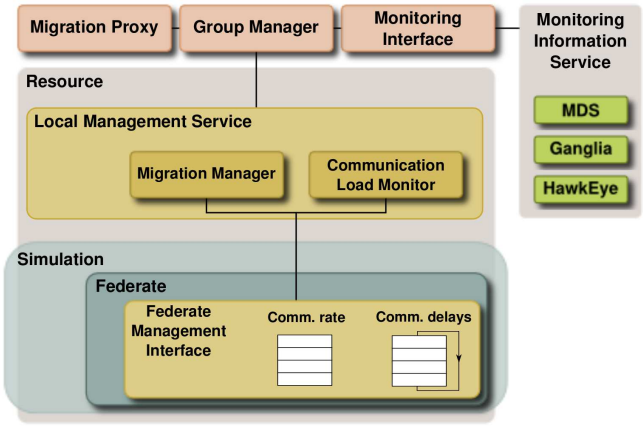


Fig. 1. The Dynamic Communication Balancing's General Architecture

a GM initiates the detection of imbalances and redistribution of load.

The Monitoring Interface is responsible for enabling the access to third-party monitoring services which provide general information about the load resource status. The monitoring information service is part of the mechanisms available in resource management systems. Conceived by Foster [28], a Grid is a resource management system that coordinates the access of individuals and institutions to distributed resources. Globus Toolkit [29] is recognized as the *de facto* middleware standard that implements the resource sharing system, and it is based on Open Grid Services Architecture (OGSA) [30]. The OGSA Grid services comprise the monitoring of resources and applications, and the scheduling and allocation of shared resources according to application requirements [31]. These services are accessed by the balancing system to retrieve the load of resources, which is gathered by the Ganglia Monitoring System.

A LMA acts as an interface between a GM and a set of federates in a resource. In order to retrieve monitoring information from federates and forward migration calls, the GM is composed of a Communication Monitoring Interface (CMI) and a Migration Manager (MM). The CMI accesses each federates load status through the Federate Management Interface (FMI). The FMI observes the communication rate of a federate, calculates the measurements, and keeps the information in a table and in a circular queue. The table stores the communication behaviour of a federate, containing the communication destination and amount of data transmitted. The circular queue includes the communication delay for each simulation interaction to a remote federate. The (MM) component forwards migration calls to the respective federate and coordinates the process of transferring a federate to a remote location.

B. Balancing Scheme

Monitoring, redistribution, and migration are the three major steps that compose the proposed balancing scheme. Monitoring requires the measurement of communication load status and communication resources conditions. The data collection is initiated in each domain by a GM, which forwards the request to its respective LMAs that subsequently triggers a CMI. Upon the request of the CMI, a FMI collects the contents of the communication table and circular queue, processes them, and answers back. The processing of communication rates consists of summing all the data transmission entries in the table since all federates communicate with the HLA RTI. Regarding the communication delay, an average is calculated from all entries in the circular queue. The CMI merges the measured metric in a message through aggregation methods and sends the message to its GM, which initiates the monitoring analysis. The GM also retrieves and filters measurements related to the computational load of resources contained in its domain through a monitoring information system.

Redistribution occurs between a GM and its neighbour GMs and produces negotiated migration calls. Even though this is the main part of the balancing scheme, it is totally dependent on reliable, efficient migrations. With receiving a migration call, a MM is responsible for efficiently transferring a federate to a predefined resource. The federate migration efficiency partially dictates the responsiveness of the balancing system since the federate transfer procedure can produce considerable cumulative delays. Consequently, a freeze-free, two-step federate migration procedure [32] [33] that minimally employs third-party mechanisms and avoids unnecessary communication and computing to be performed is adopted.

In summary, the migration technique transfers a federates code and its initialization files, suspends the federates execution, saves its execution state, and restores it at the remote destination. In the first step, a MM launches the migrating federate at the remote location to be ready for restoring its execution state with the information that is transmitted in the second migration step. In order for the federate to be launched remotely, its configuration files and any other piece of information that is needed to initialize the execution of the federate are transferred. Because reliable data transfers are required for this transfer, Grid services are used. The data transmission is performed through the staging process of GridFTP [29], which is the task realized before the submission of the migrating federate through the Web Service Grid Resource Allocation and Management (WS GRAM) [29]. All these third-party mechanisms require considerable time to perform the transfer and submission, but this time is negligible because the migrating federate does not suspend its execution.

In the second migration step, and after the federate at the remote location is ready, the communication channel of the migrating federate with the RTI is reconfigured to point to the federate at the remote host. When the channel is successfully modified, the migrating federates execution is suspended, and its running state and the state of its Local Run-time Components (LRC) are saved, transmitted, and restored at the remote federate with the mediation of a MM. The incoming messages are also saved and sent by means of the MM. These messages consist of the simulation events that are received while the federate is sus-

pended and the channel reconfiguration is performed. The messages need to be saved for the appropriate processing of the restored federate to avoid simulation inconsistencies. Finally, the federate takes over its execution, and only the delay in this second step is the final migration latency.

C. Detection and Redistribution Algorithm

The proposed detection and redistribution phase is the essential part of the balancing scheme since this phase dictates the responsiveness and efficiency of the scheme by delimiting the elements that are reallocated and their destination. Based on the placement of distributed balancing components, the characteristics and conditions of communication resources determine the relations between these components.

Delay is the metric employed in the balancing scheme to represent the dynamic conditions of communication resources. Together with the nominal network capacities, delay provides awareness of communication capabilities in a given moment. This metric indirectly supports the detection of network resource overload. By providing flexible means to perceive the dynamic communication load, delay is added to the analysis of the nominal network to extend the coordination of the inter-relations between GMs. Therefore, since communication load changes dynamically, the direction in which simulation federates are moved is modified to reflect the available resources accordingly.

In the proposed balancing scheme, communication delay is assumed as the time in milliseconds spent to transmit one byte of data. This metric is obtained by locally measuring the amount of time taken to send each message of a simulation federate. A circular queue of size N is employed as the data structure to save every delay measurement. Since communication delay presents a volatile characteristic that reflects on frequent oscillations, an average is calculated from the saved measurements to generate an estimation that reflects a more stable delay value. Size N is used to restrict the amount of information that is stored locally and to emphasize the most recent communication status. In the case that every measurement of this metric is logged during a Δ time interval, recent communication changes occurring at the end of the balancing cycle are ignored because of the large quantity of old measurements in the beginning of the interval in a long list of delay values. On the other hand, a small parameter N might not be able to represent the real communication conditions since it can produce imprecise estimations that vary excessively. Thus, as a matter of simplification, the circular queue is restricted to 200 elements; this size accommodated both low and high communication load experiments.

Not only is communication delay measured, but the total amount of data transmitted and received by a federate is collected as well. The total communication load of a federate is collected, aggregated, and sent to its GM. The aggregation is employed to decrease the amount of data transmitted between balancing components and to minimize the interference with the network resources [34]. After monitoring data is retrieved, an average delay is calculated for every resource according to the average delay of its federates. A GM also calculates the total amount of data exchanged by federates in its entire domain and the average domain delay. At the end when this information is produced, the GM requests the average communication delay and total amount of transmitted data from neighbour GMs.

Algorithm 1 Detection-Redistribution Algorithm

```
Require: delay_GM, delay_list, abs_comm_GM, abs_comm_list,
  goodput_GM, goodput_list, federate_list
  while delay_GM > max(delay_list) do
    order_ascendingly(delay_list)
    temp_GM ← select_first(delay_list)
    if delay_GM > (temp_GM.delay + error) then
      if goodput_GM ≤ temp_GM.goodput then
        data_comm ← calculate_transfer()
        repeat
          fed_eval ← select_most_comm(federate_list)
          if fed_eval.ab_comm ≤ data_comm then
            fed ← fed_eval
          end if
        until fed ≠ ∅
      else
        data_comm ← calculate_transfer()
        repeat
          fed_eval ← select_med_low(federate_list)
          if fed_eval.ab_comm ≤ data_comm then
            fed ← fed_eval
          end if
        until fed ≠ ∅
      end if
      federate_mig.add(fed)
      adjust_delays(delay_GM, temp_GM.delay)
    else
      endLoop
    end if
  end while
  GM_list ← select_similar_delay(delay_list)
  order_ascendingly(GM_list)
  for i = 1 to GM_list.size() do
    if goodput_GM > GM_list_i.goodput then
      data_comm ← calculate_transfer()
      fed_eval ← select_lowest_feds_fit(data_comm)
      if fed_eval.ab_comm ≤ data_comm then
        federate_mig ← select_lowest_feds_fit(data_comm)
      end if
      federate_mig ← fed_eval
      adjust_delays(delay_GM, temp_GM.delay)
    end if
  end for
  return federate_mig
```

As described in Algorithm 1, communication delays, amount of data transmitted, and goodput of a domain are used to identify imbalances. The goodput is a metric obtained at the application layer and represents the amount of data delivered per time. The goodput reflects the network distance of each domain from the RTI, and considers all the factors that influence the transmission time until data is delivered to the destination in its application level. Through these metrics, a greedy technique is used in the redistribution algorithm to identify a simulation distribution. The greedy analysis allows for a fast redistribution solution that offers a good load rearrangement, improving responsiveness. Therefore, based on delays, GMs are ranked, and pair analyses are performed between the GM and each neighbour GM, starting with the GM with the lowest delay.

Two actions are used in the redistribution algorithm to decrease the communication overhead. In the first action, the redistribution procedure identifies the neighbour GMs

that present communication delay plus an error lower than the GMs. This approach transfers communicative federates to locations with better network conditions by iteratively analyzing delays and goodputs. In the goodput analysis, resources with higher goodputs are assigned to receive the most communicative federates, which are ranked in a list according to their communication rate. The list is classified in high, medium, and low communication intensity. The resources not close to the RTI but with lower delay averages receive federates classified as medium communication intensity. In the second action, neighbour GMs with similar average delay ($delay_{ext} - error \leq delay \leq delay_{ext} + error$) and goodputs lower than the GMs are selected to received federates with the lowest communication intensity. In this approach, the redistribution attempts to reduce concurrency of the communication link for highly communicative federates. The value of the *error* that determines the equivalence of delays is obtained based on percentage of the local delay, and it is defined as 3%.

The next step in any of the communication delay conditions is identifying the correct federate to be migrated. This task occurs iteratively and requires an analysis of the communication load of a list of federates. The federate with the largest communication rate and less than the value obtained in Formula 1 is selected. The communication rate provides the means to assign a priority to federates. With this procedure, the balancing system prevents the production of imbalances by containing the load changes.

$$comm\_transfer = \frac{|d_2 - d_1| \times (a_1 + a_2)}{2 \times (d_1 + d_2)} \tag{1}$$

The difference of communication load between two domains used to select federates is estimated in Formula 1. The formula employs the absolute amount of data transmitted in each domain (a) and the most recent average communication delay of their GM to estimate the communication load difference. In this case, the simple subtraction is not aware of the difference between domains goodput and the external background processes. The recent communication and network pseudo-status are incorporated in the calculation through the introduction of delays. As a result, a more flexible value is achieved, which reflects the sudden network changes.

In each redistribution iteration, communication loads and delays of each GM involved with a migration move are adjusted for the next analysis iteration. The adjustment is introduced to restrict the amount of federates that are selected for migration. This restriction also prepares the analysis metrics for the next iteration. The absolute communication load value is adjusted through a subtraction ($a - a_f$) or addition ($a + a_f$), in which a is a GMs value and a_f is the communication load of a federate. For the communication delays, additional calculations are needed, which are performed through Formula 2. The formula uses the adjusted communication load value ($a - a_f$) to determine the value for the delay. Similarly, the delay of the neighbour GM is obtained through Formula 3, which proportionally considers the federate load in the domain. Then, the ranking process of neighbour GMs and the redistribution analysis are restarted.

$$adj_d = \frac{a \times d}{(a - a_f)} \tag{2}$$

$$adj_i = \frac{a \times d}{(a + a_f)} \tag{3}$$

When the load modifications are identified, federate migrations need to be issued, requiring federate identifications, source resources, and destination resources. All federate candidates belonging to the same destination domain are grouped in a list, and these federate lists are sent to their respective GMs. Upon receiving this migration request, a neighbour GM analyzes the computational load of its local resources in order to receive the federates. If computational overloads are not generated, the local resources with the least load receive the federates. Through this technique, computational imbalances are avoided while redistributing the simulation load for communication purposes. After resources are identified, the neighbour GMs send their information to the GM, which defines migration calls accordingly and forwards them to their respective federates. The resources involved with the triggered federate migrations are inserted in a list in order to be excluded from the next balancing cycle. Through this approach, enough time is provided to resources to establish their normal execution. Load irregularities (oscillations) that are caused by migrations are avoided.

IV. Extension

As exemplified in the experiments, the proposed delay-based balancing scheme presents efficiency issues related to the detection precision and the background communication load. The average communication delay in a great extent of the data sample mostly represents the current network load status. However, variations can be found in the details of this average and the majority of the variations misleads the decision-making of the balancing scheme. Imprecision originating from the communication delay oscillations in the data sample misguides the balancing scheme, which cannot identify imbalances properly or might produce unnecessary distribution modifications. The delay oscillations are caused by data transmissions with latencies that do not reflect the real conditions of the network resources, and these oscillations might exert significant influence on the monitoring metrics. Since the average communication delay is employed as the method to determine the characteristic aspect of the network resources load situation, the amount of irregular variations in the data sample proportionally misleads the final value of communication delay. Consequently, for each delay calculation, the monitoring components observe the time spent to transmit a certain amount of data, and such time can be influenced by the simulation application or by other means from the under layers in the communication system. The influences need

to be considered in order to provide a more accurate measure of communication delay. Furthermore, the other issue of the balancing scheme is the limitation of responsiveness to external background communication load. The dependency of the redistribution algorithm on the network topology hampers the redistribution actions, which have to avoid network links with large nominal bandwidth but presenting high consumption of communication resources. Therefore, a set of modifications on the redistribution algorithm is presented as an extension to improve the balancing effectiveness.

A. Extension of Data Analysis

The extension of the data analysis consists of inserting techniques and mechanisms just after data is collected or just before comparisons and interpretations are realized to determine imbalances. Essentially, these techniques comprise the exclusion of extreme values of communication delay from the circular queue and the smoothing of the GM communication average delay.

For the exclusion of non-representative values of the circular queue, the extension attempts to improve the measurement of communication delay by avoiding specific transmission situations that might influence the production of extreme values. The values are generated by a lack of precision of monitoring components in interpreting and recording the collected data. The problems with the calculated delay involves the existence of communication delays with zero value, too large transmission times for a small amount of data caused by either too little information sent or the simulation application processing during the transmission. Basically, the actions for minimizing the oscillations of delay are related to dealing with such monitoring problems.

In some cases, if the amount of data transmitted is small enough to consume less time than the precision offered by the balancing system, the monitoring mechanism records the time spent for sending data as zero. In the case of the implemented balancing system, one millisecond denotes the minimum time value detected by the local monitoring component. Consequently, the delay calculated from this transmission time is zero even though the real delay might be high. This misrepresentation leads the balancing system to incorrectly interpret the current status of the distributed system and resources. Since this technique is employed in every step of monitoring, the calculated communication delay of federates, resources, and clusters are influenced.

In order to avoid this misrepresentation of delays, an initial, simple technique is adopted. The transmission times with zero value are discarded from the recorded list since they reflect an unreal value. This value, which is generated from a lack of precision, totally differs from a real delay in any optimistic circumstance. With this action, only realistic delay values are used, providing a more representative average at the end of the calculations. Consequently, since the oscillations of the calculated average are partially caused by these discrepant values, their exclusion decreases the variations and produces a more precise representation of the recent delay.

Even with the exclusion of zero values of delay, the final average delay oscillates due to the transmission of a small amount of data. This data is large enough to produce non-zero communication delay, but its size is still small enough to make its transmission susceptible to small processing and networking time variations. In the presence of a large amount of transmitted data, these time variations are negligible since they produce insignificant changes or oscillations on the calculated communication delay. Thus, because the time variations heavily affect the transmission of a small amount of data and cannot be avoided, communication delays for these transmissions are not considered in the calculation of the federate average delay. In the proposed extended scheme, 100 bytes is the minimum amount of transmitted data to be considered in the calculation of delays, and every delay based on a quantity of data inferior to this amount is discarded from the circular queue.

The detection and exclusion of high, discrepant communication delay values is another measure adopted in the extension to decrease the communication delay oscillations. Even avoiding messages with zero transmission time and with too small of a payload, object updates that fit in a category which is considered normal might produce delays that differ totally from the rest of the delays in the data sample. These discrepant values are originated from ongoing processing that occurs simultaneously with the data transmission and hampers the measurement of the sending time. Also, sudden spikes of communication resources consumption by other processes considerably influences the simulation communication delays. These spikes reflect on a few data transmissions, but they do not represent the general communication conditions in certain locations of the distributed environment. Because these sudden changes might happen unpredictably, the discrepant values need to be detected and eliminated from the data sample to avoid erroneous calculation of the final average communication delay. The detection is performed after the delays are calculated and when they are requested by the balancing system for redistribution analysis through the LMA. The detection and exclusion is postponed to the last moment before all elements in the circular queue are retrieved, so the analysis of the data sample for detection can identify the divergent values based on the data sample pattern. The analysis consists of using the interquartile range to determine and eliminate the outliers in the data sample. Normally, the collected communication delays tend to present a gaussian distribution for a large data sample, and the existence of outliers may influence the average delay to differ from its real normal distribution. The interquartile range presents a robust statistical method with a breakdown point of 25%, providing an efficient technique for identifying outliers for exclusion and improving the representation accuracy of the average. After the discrepant values are identified and eliminated, the average communication delay is calculated based on the middle fifty values of the sample data.

Even though the filtering of gathered delay metrics considerably decreases the sharp oscillations of average communication delay, smoothing techniques are applied on the GM averages to diminish the inappropriate variations between them and to determine a trend. The sequence of the GM communication delay averages is composed of a list of successive data points naturally ordered according to their time and spaced at uniform time intervals. Due to this natural temporal ordering characteristic, this sequence of delays is observed and interpreted as a time series that oscillates (decreases and increases) along with the time. Techniques for processing time series can be applied to these averages in order to produce a prediction or identify an smoothed value. One of these techniques that can be used for determining a more steady delay value based on the past oscillations is exponential smoothing. In this smoothing method, weights are assigned to each data point in the time sequence and according to time distance from the current time. Exponential smoothing acts as an iterative simple weighed average calculation based on the previous observation and the previous smoothed average. Even though single exponential smoothing offers a fair smoothing for a time series, it does not consider trends in the data sequence. Such trends present high relevance for this smoothing analysis since they can be identified in the load oscillations of the data sample. In order to incorporate the trend of the communication delays, double exponential smoothing is employed.

The double exponential smoothing technique employed in the GMs communication delay analysis is represented by Formulas 4 and 5. These associated formulas are used when the average is requested for the redistribution of a neighbour GM, as presented in Algorithm 1. The local GM, upon the request of its neighbour GM, generates the smoothed delay average instead of the pure delay average and returns it back to the requester. Smoothing is applied among the current and the past communication delay averages calculated for the GMs. Observing that the current value of the sequence of communication delays is used to calculate its smoothed value replacement in double exponential smoothing, a GM does not need to save all the past data (average delays) since these are reflected in the parameters for the calculation of the last exponential smoothing.

$$sum_i = \alpha \times elem_i + (1-\alpha) \times (sum_{i-1} + t_{i-1}), 0 \leq \alpha \leq 1 \quad (4)$$

$$t_i = \gamma \times (sum_i - sum_{i-1}) + (1-\gamma) \times t_{i-1}, 0 \leq \gamma \leq 1 \quad (5)$$

The smoothing technique is only employed at a data sample for i larger than 1 since it needs to have at least one previous element in the data sequence to support the current element. One simple method of initialization of the smoothing function is to assign the first average obtained to the first smoothed valued, so sum_0 just receives $elem_0$. Both functions are configured through two constants: the smoothing function is defined by the smoothing constant α, which works in conjunction with γ that delimits the trend estimation function. These values are chosen according to application-specific needs, and for this extension, α is set to 0.9 and γ is set to 0.5. Since α is set to a high value in the proposed extension of the balancing scheme, the initialization elements do not present much influence on the final smoothed average but they exercise certain weight to generate smoothing. For each calculation, a GM saves the sum_i and the t_i to use for the next average request of its average.

B. Extension of Load Redistribution

The redistribution algorithm also requires modifications in order to consider the existence of external background communication load in the system. In the previous algorithm, the presence of an external process might substantially influence the communication conditions for a distributed simulation, so the balancing scheme needs to consider communication delay more heavily in the redistribution. The former redistribution algorithm observes delay and uses goodput to orientate its load reallocation. In the presence of external background load, the delay based balancing scheme slowly redistributes the load by transferring the most communicative federates at the end due to the goodput constraints. However, the goodput cannot be discarded since it is the only parameter that provides a further view of the environment resources. In this case, the redistribution algorithm is extended by introducing an additional condition that attempts to improve the responsiveness of the balancing scheme to external background communication load, as described in Algorithm 2. The added condition is based on a threshold that identifies a large difference of communication delay between domains. The threshold ext_resp is a percentage of a neighbours delay and is incorporated to the neighbour GMs communication delay for the analytical comparisons. The threshold is defined proportionally to the difference of a GMs goodput ($ext\_resp = temp\_GM.delay \times (goodput_{GM} - temp\_GM.goodput)/temp\_GM.goodput$ when the $goodput_{GM} > temp\_GM.goodput$ and $ext\_resp = error$ otherwise).

V. Experimental Results

Experiments have been conducted to analyze and compare the proposed redistribution schemes effectiveness with the previous distributed balancing system [4]. The experimental analysis consisted of the execution of HLA-based distributed virtual simulations deployed on two computing clusters and one computing server. One of the clusters (IBM) was composed of 24 computing servers, which were inter-connected through a gigabit Ethernet network link. Each computing server contained a Quadcore 2.40GHz Intel CPU and 8 gigabytes of RAM. The other cluster (Dell) consisted of 32 computing servers that were connected through a 2-gigabit Myrinet optical network link. Each computing node contained a Core 2 Duo 3.4 GHz Intel

Algorithm 2 Extended Redistribution Algorithm

```
Require: delay_GM, delay_list, abs_comm_GM, abs_comm_list,
  goodput_GM, goodput_list, federate_list
  while delay_GM > max(delay_list) do
    order_ascendingly(delay_list)
    temp_GM ← select_first(delay_list)
    if delay_GM > (temp_GM.delay+) then
      data_comm ← calculate_transfer()
      repeat
        fed_eval ← select_most_comm(federate_list)
        if fed_eval.ab_comm ≤ data_comm then
          fed ← fed_eval
        end if
      until fed ≠ ∅
      federate_mig.add(fed)
      adjust_delays(delay_GM, temp_GM.delay)
    else if delay_GM > (temp_GM.delay + error) then
      if goodput_GM ≤ temp_GM.goodput then
        data_comm ← calculate_transfer()
        repeat
          fed_eval ← select_most_comm(federate_list)
          if fed_eval.ab_comm ≤ data_comm then
            fed ← fed_eval
          end if
        until fed ≠ ∅
      else
        data_comm ← calculate_transfer()
        repeat
          fed_eval ← select_med_low(federate_list)
          if fed_eval.ab_comm ≤ data_comm then
            fed ← fed_eval
          end if
        until fed ≠ ∅
      end if
      federate_mig.add(fed)
      adjust_delays(delay_GM, temp_GM.delay)
    else
      endLoop
    end if
  end while
  GM_list ← select_similar_delay(delay_list)
  order_ascendingly(GM_list)
  for i = 1 to GM_list.size() do
    if goodput_GM > GM_list_i.goodput then
      data_comm ← calculate_transfer()
      fed_eval ← select_lowest_feds_fit(data_comm)
      if fed_eval.ab_comm ≤ data_comm then
        federate_mig ← select_lowest_feds_fit(data_comm)
      end if
      federate_mig ← fed_eval
      adjust_delays(delay_GM, temp_GM.delay)
    end if
  end for
  return federate_mig
```

CPU and 2 gigabytes of RAM. The additional computing server contained an Intel i7 CPU with 8 cores and 6 gigabytes of RAM. This external computing server was located in a different network, so it was connected to the Dell cluster through a fast Ethernet network link and to the IBM cluster through either a 10 base T network link or a fast Ethernet network link. In this environment, a Linux operating system and the Globus Toolkit 4.2.1 [29] were installed to support the experiments, and the HLA platform with RTI version 1.3 was used to develop and coordinate the virtual simulations.

All the simulation federates were evenly placed on the 56 computing servers of both clusters. The HLA RTI executive was deployed on the external computing server and coordinated all the experimental simulations. The proposed balancing systems were also placed on each computing server of both clusters. A LMA was deployed in each cluster computing server, and a GM was running in each cluster management node.

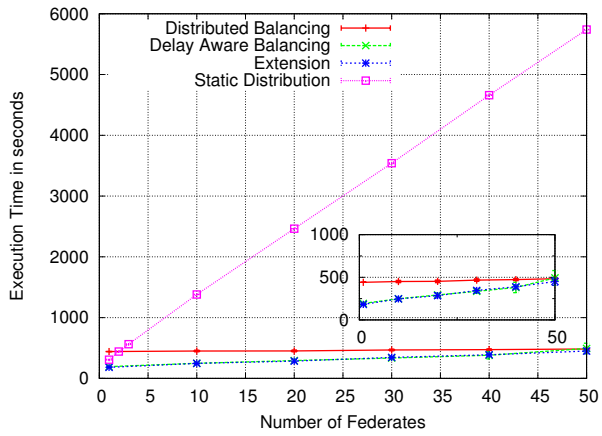
For the simulation scenario used in the experiments, the federates organized simulations that performed training operations coordinated in a two-dimension routing space. In such scenarios, two teams of interactive tanks were controlled by federates in time-stepped simulations. Each federate calculated the movement of a set of tanks, and for communication purposes, the tanks performed simple movements that did not require highly intensive computing. Each federate also realized the publication of new positions and subscription to other federates tank updates. Generally, 200 federates composed the simulations and organized 1 to 10 tanks each in 100 time steps. In order to control the communication imbalances, the tanks updates were reduced to around 700 bytes while special objects were introduced in some federates to apply a 54000-byte load, producing considerable communication overhead.

The experiments were divided in two study groups. In one group, the 10 base T network connection dictated the simulation performance due to the simulation dependency characteristics. The connection with the smaller bandwidth intentionally generated a bottleneck for the distributed simulation when it was employed in the experiments. In the other group, an external background communication load was inserted in the communication link between the management node of the Dell cluster to generate a communication imbalance. The load consisted of a highly intensive consumption of network resources by running a client process that constantly transmitted and received large amounts of data to a server process.

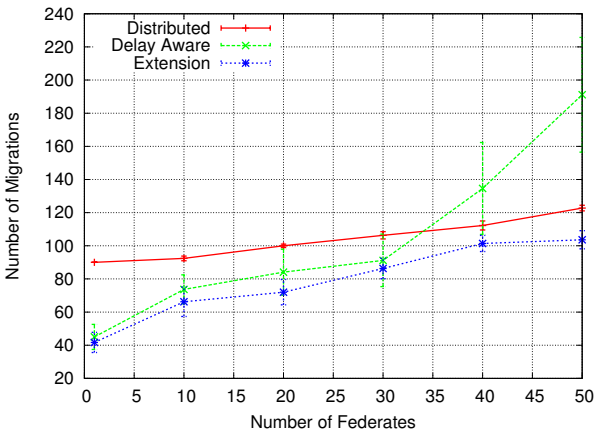
A. Imbalances Caused by Network Resources

A static, restricted communication imbalance was introduced in the experimental simulations in order to observe the responsiveness of the proposed balancing schemes. As shown in Figure 2(a), every simulation federate controlled one object (tank). Some federates were selected to additionally coordinate the update of special objects, which generated considerable communication overload in the system. The curves in the graph show the influence of the communication delay on simulation performance as the number of communicative federates increased in the simulations. The static distribution presented the worst simulation performance with the induced communication imbalance since it did not modify the distribution according to the simulation communication behaviour. This linear, fast execution time increase was a consequence of the severe bottleneck introduced between the IBM cluster and the external computing server. All the proposed and distributed balancing systems presented curves that grew similarly. However, observing the inset graph in Figure 2(a), the proposed schemes showed an improvement, which achieved almost double the performance improvement when compared with the distributed approach for some cases.

For the same simulations, Figure 2(b) describes the number of migrations required by each balancing scheme to obtain a simulation performance improvement. The proposed balancing system presented less migrations than the distributed balancing system for simulations with up to

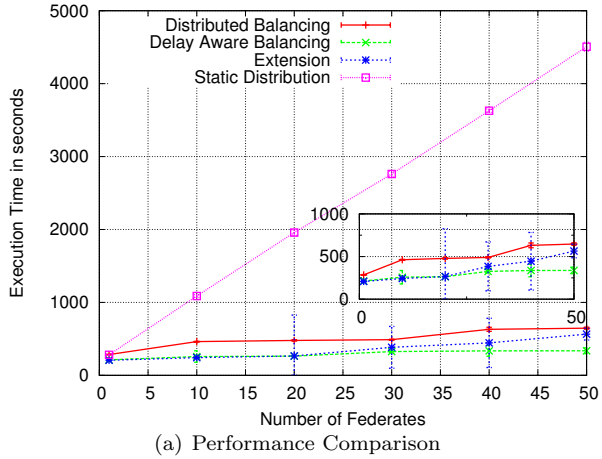


(a) Performance Comparison

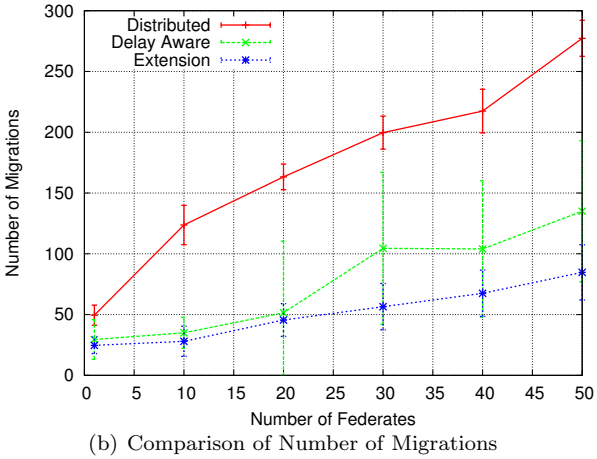


(b) Comparison of Number of Migrations

Fig. 2. Balancing Scheme Analysis for Increasing Amount of Static Communication Load in Presence of a Network Imbalance



(a) Performance Comparison



(b) Comparison of Number of Migrations

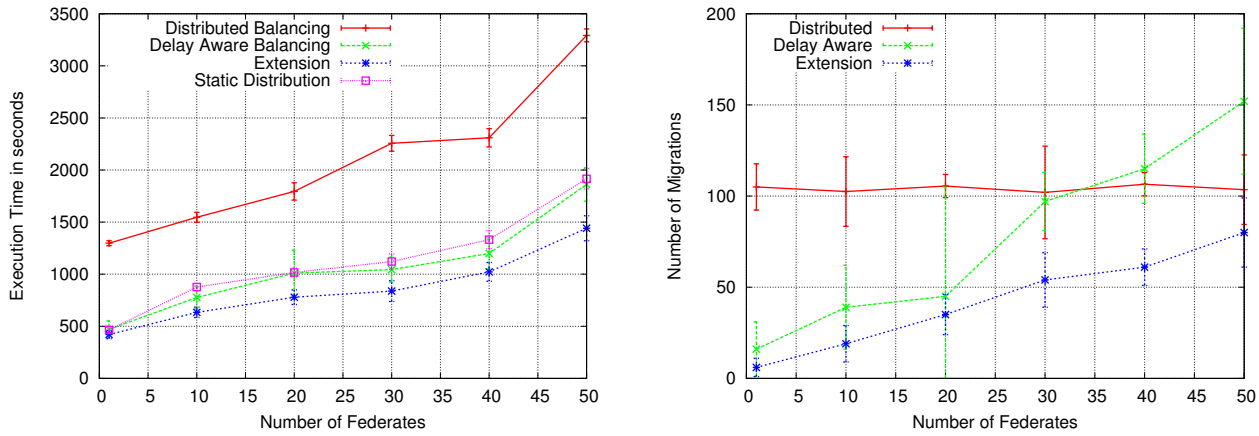
Fig. 3. Balancing Scheme Analysis for Increasing Amount of Dynamic Communication Load in Presence of a Network Imbalance

30 federates. This steep increase of the number of migrations originates with the instability of the proposed scheme and is caused by the measured communication delay oscillations. When observing the results of the extension approach, there was a significant decrease in the number of migrations with the improvement of the precision of the balancing. The large amount of migrations of the distributed approach was a result of the inter-relations between balancing components in the distributed scheme that constantly moved federates between domains, and a large effort was spent to achieve such a distribution. Even with this simulation scenario that does not impose large peer-to-peer data transfers for federate migrations, the large amount of migrations noticeably influenced the final simulation performance result.

In this experiment, the same static, restricted communication imbalance was imposed to the distributed virtual simulations, but the simulations generated a communication load that changed dynamically. In this case, the responsiveness of the balancing schemes was analyzed since the dynamic imbalances require fast, effective redistribution of simulation elements. As depicted in Figure 3(a), the proposed balancing schemes were more effective in reducing the simulation execution time, and the extended scheme was able to outperform the delay-based redistribution for simulations with more than 20 communicative federates. The extension also showed a reduced amount of oscillations when compared to the regular scheme, as described by the standard deviation of the curves. These oscillations are more evident in the curves that represent the number of balancing migrations in Figure 3(b). According to the curves, both proposed schemes increased the balancing efficiency, since they reduced the amount of migrations to achieve better performance gain. Nevertheless, when comparing the extended and regular approaches, the extended approach presented significantly smaller variations in its curve. This conforms with the decrease of communication delay oscillations and consequently the improvement of balancing precision.

B. Imbalances Caused by External Communication Load

An experimental scenario containing external background communication load was produced in order to analyze the responsiveness of the proposed balancing schemes



(a) Performance Comparison (b) Comparison of Number of Migrations

Fig. 4. Balancing Scheme Analysis for Increasing Amount of Static Communication Load in Presence of External Background Overhead

when compared to the distributed balancing scheme. In this scenario, both clusters were connected to the external computing server through the same fast Ethernet network link. In this case, the Dell cluster presented slightly better communication conditions since it contains a Myrinet optical switch connecting its internal computing servers. However, a process was placed in one of the computing servers of the Dell cluster in order to produce excessive communication load. This load was applied for λ amount of time (180 seconds) and then was kept deactivated for γ (40 seconds). This activation and deactivation behaviour, together with the dynamic simulation load changes, was introduced in the scenario as an attempt to mimic the unpredictability of a real shared network, in which any application might be consuming the resources at any moment. Thus, the communication load dynamically generated imbalances and overhead for the distributed simulations, and the reaction of the balancing schemes were observed.

As shown in Figure 4(a), the arrangement of simulation load defined by the distributed balancing approach presented a worse simulation execution time than the simulations with just the static deployment. This situation was generated by the unawareness of the distributed scheme of the external background load of this balancing system. The external background process produced some communication overhead that stressed the network, but the situation was worsened when the distributed balancing placed the most communicative federates in the network bottleneck. The delay-based balancing scheme did not present much performance gain to the simulations since the communication delay oscillations misled the balancing scheme to generate too many migrations. The extended scheme was able to decrease the simulation time, but it offered only a slight performance gain. The curves in Figure 4(b) fairly represented the unawareness of the distributed scheme, which did not modify its reaction to the external communication load. The regular proposed scheme showed a growing, varying amount of migrations as the number of federates increased; this reflects the instability of the balancing due to the variations in its metrics.

VI. Conclusion

A delay-based redistribution scheme and an extension are proposed to coordinate the reallocation of distributed load for HLA-based virtual simulations. Their balancing elements are organized in a hierarchical structure, which is built according to the network topology of the environment used to deploy the distributed simulations. The proposed schemes rely on the measurement of simulation and resources, analysis, detection, redistribution, and migration. The introduction of communication delay as major metric in the schemes requires modifications to the inter-relations between balancing components to allow independent, asynchronous interactions. The proposed extension introduces filtering and processing techniques for the measured communication delay to decrease the delay oscillations that interfere with the efficiency of the balancing system. The extension also modifies the redistribution algorithm to increase the responsiveness of the balancing system to imbalances caused by external background communication load.

Experiments have been conducted to evaluate the performance gain and the balancing efficiency of the proposed balancing scheme and its extension when compared with a balancing system that presents a distributed reallocation scheme. In the overall experimental analysis, both proposed balancing techniques reduce the number of costly migrations and provide a method of measuring the actual status load of communication resources. Moreover, the proposed extension improved the precision of the delay-based balancing scheme by decreasing the oscillations of the measured metrics. This increase of accuracy allowed the balancing system to avoid unnecessary precipitated load transfers, reducing the number of migrations. As future work, additional experimental analyses will be performed for other realistic scenarios in order to evaluate the responsiveness of the proposed balancing schemes. The number of migrations directly impacted on the execution time of the simulations, so additional studies will be conducted to determine the influence of migration latency in the balancing system. Due to the importance of balanc-

ing the computational load, it will be incorporated in the communication balancing scheme to provide a broader load redistribution solution for distributed virtual simulations.

Acknowledgement

This work is partially supported by NSERC, the Canada Research Chair program, MRI/ORF research funds, the Ontario Distinguished Research Award, and the EAR Research Award.

References

[1] "Ieee standard for modeling and simulation (m&s) high level architecture (hla) framework and rules," IEEE Standard 1516-2000, September 2000.

[2] R. E. De Grande, A. Boukerche, and H. M. S. Ramadan, "Decreasing communication latency through dynamic measurement, analysis, and partitioning for distributed virtual simulations," *Instrumentation and Measurement, IEEE Transactions on*, vol. PP, no. 99, pp. 1–12, 2010.

[3] R. E. De Grande and A. Boukerche, "Dynamic partitioning of distributed virtual simulations for reducing communication load," in *Proceedings of the IEEE International Workshop on Haptic Audio visual Environments and Games (HAVE)*. 2009, pp. 176–181, IEEE Computer Society.

[4] R. E. De Grande and A. Boukerche, "Distributed dynamic balancing of communication load for large-scale hla-based simulations," in *Proceedings of the The IEEE symposium on Computers and Communications*, Washington, DC, USA, 2010, ISCC '10, pp. 1109–1114, IEEE Computer Society.

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

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

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

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

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

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

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

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

[13] 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.

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

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

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

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

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

[19] 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*. 1999, pp. 101–108, IEEE Computer Society.

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

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

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

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

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

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

[26] 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.

[27] R. E. De Grande and A. Boukerche, "A redistribution scheme centred on communication delay for distributed virtual simulations," in *Proceedings of the IEEE International Workshop on Haptic Audio visual Environments and Games (HAVE)*. 2010, pp. 45–50, IEEE Computer Society.

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

[29] "Globus," University of Chicago, 7 Feb. 2008.

[30] 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.

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

[32] A. Boukerche and R. E. De Grande, "Optimized federate migration for large-scale hla-based simulations," in *Proceedings of the 12th International Symposium on Distributed Simulation and Real-Time Applications*. 2008, pp. 227–235, IEEE Computer Society.

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

[34] Ivan Rodero, Francesc Guim, Julita Corbalan, Liana Fong, and S. Masoud Sadjadi, "Grid broker selection strategies using aggregated resource information," *Future Generation Computer Systems*, vol. 26, pp. 72–86, 2010.