Towards Grid-Wide Modeling and Simulation

Y. Xie[1], Y.M. Teo[1,2], W. Cai[3] and S.J. Turner[3]
[1]Singapore-Massachusetts Institute of Technology Alliance, [2]National University of Singapore
[3]Nanyang Technological University

***Abstract*— Modeling and simulation permeate all areas of business, science and engineering. With the increase in the scale and complexity of simulations, large amounts of computational resources are required, and collaborative model development is needed, as multiple parties could be involved in the development process. The Grid provides a platform for coordinated resource sharing and application development and execution. In this paper, we survey existing technologies in modeling and simulation, and we focus on interoperability and composability of simulation components for both simulation development and execution. We also present our recent work on an HLA-based simulation framework on the Grid, and discuss the issues to achieve composability.**

***Index Terms*— Modeling and simulation, Grid computing, interoperability, composability.**

I. Introduction

MODELING and simulation provide a low cost and safe alternative to real-world training, experiments, analysis of natural phenomena, etc. The High Level Architecture (HLA), approved by US Department of Defense (DoD) in 1995, provides an architecture for interoperability and reuse in distributed simulation [16], and it was adopted as an open standard through the IEEE Standard 1516 in September 2000.

A simulation, or a *federation* in HLA's terminology, consists of a set of logically related simulators, called *federates*. Federates communicate with each other through the *Run-Time Infrastructure* (RTI). HLA defines the rules and specifications to support reusability and interoperability amongst the simulation federates. The RTI software supports and synchronizes the interactions amongst different federates that conform to the standard HLA specification [16].

With the increase in the complexity of simulations, large amounts of computational resources are needed, and collaborative model development is needed, as multiple parties could be involved in the development process. Interoperability and composability of simulators have become two of the most important factors in simulation model development and execution [17], [34].

- *Interoperability* refers to the capability that components, i.e. models in our case, can communicate with each other meaningfully. This requires standard ways to develop models not only at the syntax level, but also how semantic information can be shared among models. The underlying infrastructure should also support such communications. Moreover, other practical issues are involved as well, such as human factors, security, etc.
- *Composability* deals with the selection and assembly of shared models in various combinations into a complete and validated environment to satisfy user requirements meaningfully. Model developers should be able to reuse existing shared models as sub-models to form composite models collaboratively using the infrastructure. When models are being executed, individual sub-models' constraints such as quality of service (QoS) have to be satisfied to make the composite models work properly. When certain sub-models' constraints cannot be met during execution, they should be replaced by other sub-models dynamically. To achieve composability, interoperability is a must.

The concept of "Grid" computing was proposed by Ian Foster as secure and coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [20]. Much effort has been made in the area of Grid computing since 2001. A number of national and international Grid infrastructures have been set up, such as [6], [13]. Many systems and middleware for Grid computing have been proposed in the past few years, and Globus [21] is becoming the de facto standard middleware for Grid computing. The third version of the Globus Toolkit [23] is based on the concept of *Grid Services*, which is defined by the Open Grid Services Architecture (OGSA) [22], and is specified by the Open Grid Services Infrastructure (OGSI) [36]. The Grid provides a platform for scalable and coordinated resource sharing and collaborative application development and execution.

In this paper, we first review the fundamentals of HLA-based distributed simulation in Section II. Then, we survey existing technologies in modeling and simulation, and focus on interoperability and composability of simulation components for both development and execution in Section III and IV respectively. In Section V, we present a general architecture and our recent work on an HLA-based simulation framework on the Grid, called HLAGrid, to support model execution. The design and preliminary experimental results of the HLAGrid are described in Section VI. In Section VII, we draw brief conclusions and discuss future work.

II. HLA-based Distributed Simulation

In HLA-based distributed simulations, the Run-Time Infrastructure (RTI) provides common services during the execution

Y. Xie is with the Singapore-Massachusetts Institute of Technology Alliance. Email: smaxy@nus.edu.sg

Y.M. Teo is with the Singapore-Massachusetts Institute of Technology Alliance and Department of Computer Science, National University of Singapore. Email: teoym@comp.nus.edu.sg

W. Cai and S.J. Turner are with the School of Computer Engineering, Nanyang Technological University. Email: {aswtcai,ASSJTurner}@ntu.edu.sg

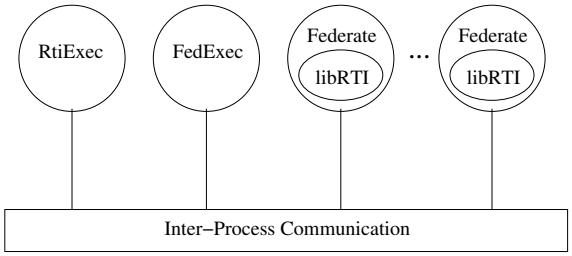

Fig. 1. RTI components.

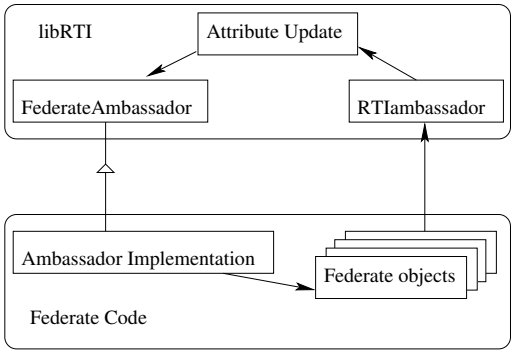

Fig. 2. Federate code communicate with libRTI through method calls of *RTIambassador* and the callback functions provided in the *FederateAmbassador* class.

of a HLA simulation by implementing the HLA interface specification [16]. The RTI consists of the RTI Executive process (RtiExec), Federation Executive process (FedExec) and the libRTI library, as illustrated in Figure 1.

The RtiExec manages multiple federation executions with different names within a network. The FedExec manages multiple federates within a federation execution, and there is one FedExec instance per federation. LibRTI is a library that provides HLA services to federates. More specifically, all requests made by a federate on the RTI take the form of a method call to the object *RTIambassador* within libRTI, and the federate code should implement the abstract class *FederateAmbassador* to provide callbacks functions. Figure 2 shows how federate codes communicate with the libRTI.

HLA uses the object oriented model, in which each federate has objects, which are instances of classes with attributes, and are specified in the Simulation Object Model (SOM). Data can be communicated between federates via the change in attributes of objects that federates own. The federate who offers the attributes to others *publishes* the attributes and the federates that need the attributes *subscribe* to the attributes. Another way for inter-federate communication is in the form of *interaction*, which is a transient communication and is not stored.

III. Interoperability

Interoperability of systems is a complex issue which involves two main aspects: conceptual and practical. From the conceptual point of view, interoperability must be *meaningful*: the system should support component-based development. From the practical point of view, interoperability must be *manageable*: the system should enable components to be used in a secure, flexible and coordinated manner.

We extend the five-level model for conceptual interoperability described in [35], and propose a two-dimensional model, shown in Figure 3.

The vertical axis consists of five layers of conceptual interoperability [35]:

- *Level* 0 - *System Specific data*: data is used in a proprietary manner.
- *Level* 1 - *Documented data*: data is documented in a template or protocol, such as HLA's Object Model Template.
- *Level* 2 - *Aligned Static data*: data is documented using a static reference model based on a common ontology.
- *Level* 3 - *Aligned Dynamic data*: data is used within the component as defined by standard methods such as UML.
- *Level* 4 - *Harmonized data*: besides the implementation and documentation, additional information are required to achieve interoperability.

Though simulation components can be conceptually interoperable, they may not work with each other in reality, because of security policy, human factors and so on. We introduce three levels on the horizontal axis:

- *Level* 0 - *Cluster*: components are within a single cluster, and they are managed by a single administrator.
- *Level* 1 - *Organization*: components could be physically located across the internet, and they are managed under different cluster-administrators, but they typically belong to the same organization with an overall administrator who coordinates all the cluster-administrators in the organization.
- *Level* 2 - *Grid*: components are managed by different administration domains, and they belong to different virtual organizations (VOs) [20].

There has been some progress along the vertical axis to build more conceptually interoperable systems for distributed simulations, such as the HLA, which belong to Level 1 "Documented data", and several other proposed projects [3], which belongs to level 2 for conceptual interoperability. However, the effect of practical interoperability along the horizontal line has been largely neglected, with not enough attention paid to it in the design of earlier systems, such as the HLA. It can be demonstrated from many industrial applications that the HLA is very successful at the level of "Organization", yet very little success has been shown about how HLA could support the level of "Grid". Besides various technical obstacles, there are several important practical factors which restrict the interoperability of existing HLA applications:

- *Firewall*: HLA requires specific ports to be open for federates to communicate with the RTI and other federates, which may be very difficult or even impossible at the Grid level.
- *Multicasting*: in HLA simulation applications where a significant amount of multicast streaming is involved, the current internet infrastructure may not be sufficient, because Internet Service Providers (ISPs) may not enable multicast at the IP level, which is out of the range of simulation policy. There is some ongoing research [29]

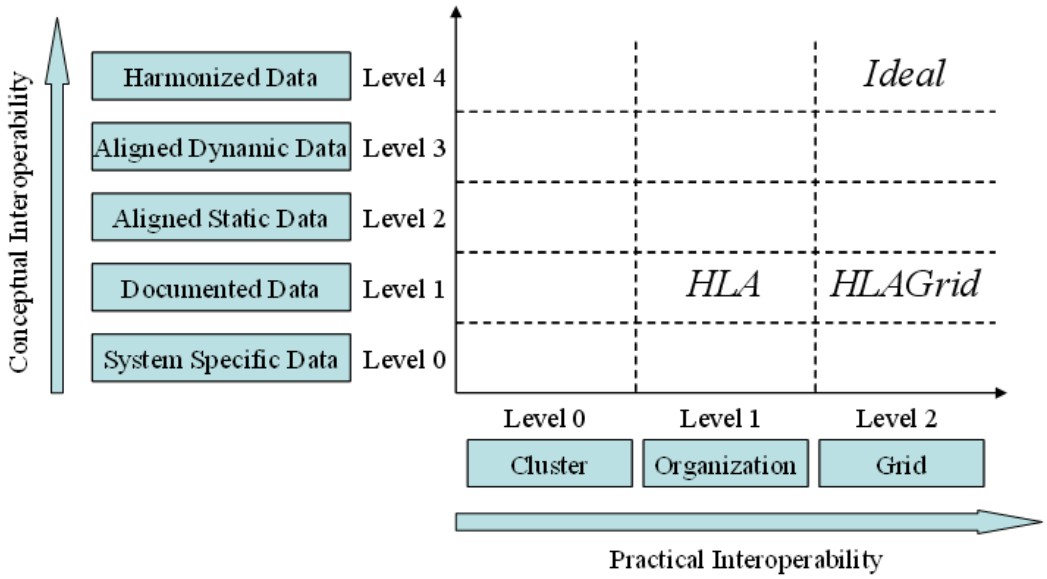

Fig. 3. Two-dimensional model for Interoperability.

to address this issue.

- *Distributed Denial of Service*: similar to the multicasting issue, ISPs have low incentive to adopt any anti-DDoS measures to protect federates and RTIs.
- *Trust*: no trust model exists for HLA to allow different virtual organizations to collaborate on simulations. The impact of lacking trust between the organizations would simply prevent collaborations from happening.
- *Accounting*: compared with the organization case, the Grid requires more coordination among domains for accounting.

Issues along the horizontal axis cannot be resolved solely by advances in technology, because they fundamentally require common understanding and management policy to be established to allow any technically-sound interoperable components to be practical. Likewise, the next phase of the Semantic Web [12] holds the promise that applications can more readily interact without human involvement, but there is still a gap for a technically-viable semantic web to be deployed in practice, which involves human coordination and management issues.

IV. Composability

Composability is the capability to select and assemble shared components in various combinations into a complete and validated environment to satisfy specific user requirements meaningfully. A component includes not only the software, but also a precise specification of the functionality it provides with all dependencies (hardware, software, version and other components). Composability is desirable, because it makes creation of a composite system easier by reusing existing models, it helps in understanding complex systems, and it simplifies the testing and maintenance of systems as well [17]. But, composability is difficult to achieve, because it requires efforts in both the modeling itself and the development of the infrastructure where models are being composed and used.

- *Modeling*: The system to be modeled could be very complex. Depending on the objective of the modeling (e.g. analysis, daily-routine simulation), the process of modeling with composability could be very time consuming, especially for modeling of large scale systems.
- *Infrastructure*: Many factors of the infrastructure are involved: network, fault tolerance, human factors, etc.

Recently, the Model-Driven Architecture (MDA) [32] has been proposed to promote the concept of a common stable meta-model, which is language-, platform-, and vendor-neutral. The main idea is that even if the underlying infrastructure shifts over time, the meta-model remains stable, and the only changes are the tools to bridge the meta-model and infrastructure. The core of the MDA includes SOAP, UML, XML with related XML Metadata Interchange (XMI) specification, and so on.

Tolk [34] proposed a framework to merge the existing simulation standard, the HLA, into the MDA, and the potential of composing simulation models is huge. A thorough study of composability of modeling and simulation in the context of defense systems is presented in [17], and many research issues have been pointed out.

In the web service community, the issue of composition has received a lot of attention as well. Grønmo, et al. [25] propose a model-driven web service development process, where web service descriptions are imported into UML models; integrated into composite web services; and the new web service descriptions are exported. Liang, et al. [27] propose a semi-automatic approach to composite web service discovery, description and invocation. Their main approach makes use of a registry with constraint matching capabilities to support composite service discovery and description. A user interface is provided for interactive composing of a service request, and a search algorithm is used to construct a composite service template. A composite service processor is designed to execute composite services by invoking the component service operations of

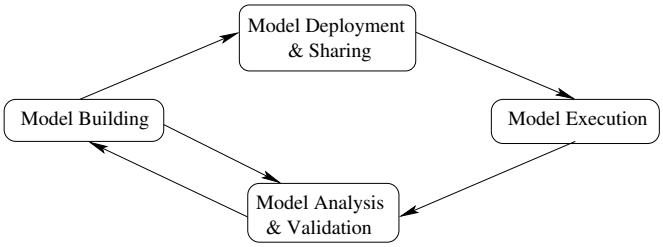

Fig. 4. Life cycle of simulation models.

various service providers. In [14], [26], a composite service is modeled as a structure of "sub-services" using a service flow language such as the Web Service Flow Language and the Business Process Execution Language, but these compositions are manual. Some ongoing research efforts use rule systems to deduce a composite service from register services [33] dynamically. There are also ongoing efforts [15], [28] from the AI community, and they interpret service composition as a planning or reasoning task with its execution.

V. Grid-Wide Framework for Modeling and Simulation

A. Overview

The Grid-wide framework for modeling and simulation should support the whole life cycle of any simulation model, which involves model development, deployment, sharing, execution, analysis and validation. Figure 4 shows the relationship between the four main steps in the life cycle of simulation models.

The *building*, and *analysis & validation* steps are simulation specific, while the *deployment & sharing* and *execution* is dependent on the underlying infrastructure.

B. Current Work

Our recent work focuses on the interoperability issue and present a distributed simulation framework to extend the practical interoperability of the existing HLA from "organization" to "Grid". The framework addresses several important issues involved in the extension of the HLA, such as heterogeneity, federation discovery, security, and performance. The framework achieves interoperability between different simulators (federates) by using a Federate-Proxy-RTI architecture, in which a remote proxy acts on behalf of the federate in interacting with the RTI. It hides the heterogeneity of the simulators, simulators' execution platforms, and how the simulators communicate with the RTI. Moreover, different RTI services can be exposed as Grid services, which provides more secure, scalable and coordinated management. RTI services' internal data are exposed as Grid service data elements, which allows both *pull* and *push* kind of access by other Grid services. All interfaces used in the framework comply with the standard HLA interface specification, which provides reusability to simulators. A prototype of the framework is implemented using DMSO's RTI 1.3NG version 6 and the Globus Toolkit Version 3. The DMSO HLA's benchmark programs have been converted from C++ to Java, and are used in the testing of the prototype.

C. Extending HLA's Practical Interoperability

The interoperability model proposed has set the context of how modeling and simulation should be carried out in an interoperable way. This paper focuses on upgrading HLA's practical interoperability from "organization" to "Grid". Issues that need to be addressed include:

- *Heterogeneity*: Heterogeneity of simulators, simulators' execution platforms, how simulators communicate with the RTI should be considered.
- *Federation Discovery*: An indexing service for locating the RTI and federation needs to be provided on the Grid. The existing HLA does not support dynamic discovery of the federation.
- *Security*: The simulation model should run at the client side, whereas other services should provide simulation management such as federation management, time management, etc.
- *Performance*: Communication over the Grid could incur overhead due to latency of the network, so performance should also be considered to justify the tradeoff of the Grid-enabled HLA.

D. Related Work

In recent literature on the HLA, some researchers [24], [37] focus on building tools to ease the process of modeling and simulation using HLA, and some [38] concentrate on adding auxiliary systems to make HLA more useful in a wide-area-network (WAN), while others [8], [19], [34], [35] have a more ambitious objective to reinvent HLA to be model-driven and composable, or replace HLA with a new framework, etc. The proposed model for interoperability shows how much improvement in interoperability a particular work has achieved, and how a particular work relates to others' work. The model for interoperability proposed gives an overall picture of the work that has been done and projects possible avenues for future work.

Some researchers focus on the extension of conceptual interoperability, such as [34], [35], in which Tolk, et al. propose a framework to integrate HLA into the Model-Driven Architecture (MDA) [32] defined by the Object Management Group (OMG) to improve interoperability.

Research into improving the practical interoperability includes [19], [30], [37], [38] . Wytzisk, et al. [37] propose a solution that brings HLA and the Open GIS Consortium (OGC) [10] standard together. It provides external initialization of federations, controlled start up and termination of federates, interactions with running federates, and access of simulation results by external processes. Zajac, et al. [38] propose a system to enable HLA-based simulations on the Grid, but they focus on migrating federates. The system also includes discovery, information indexing services, etc. However, the communication between RTI and federates is based on the original HLA communication, which requires predefined ports to be open. Fitzgibbons, et al. [19] present a distributed simulation framework, called IDSim, based upon OGSI [36]. It makes use of Globus's Grid service data elements as simulation states to allow both pull and push

modes of access. It also aims to ease the integration and deployment of tasks through inheritance. Moreover, the distributed simulation communities initiated a plan, called the Extensible Modeling and Simulation Framework (XMSF) [8], which defines a set of Web-based technologies, applied within an extensible framework, that enables a new generation of modeling and simulation applications to emerge, develop and interoperate. Morse, et al. [30] propose an architecture using Web Services for web based federates communicating with the RTI. Their approach is based on formatting the RTI calls via Simple Object Access Protocol (SOAP) [2] and employing the Blocks Extensible Exchange Protocol (BEEP) [1] communication layer to enable bi-directional calls and callbacks via web services. None of these addresses all the issues discussed in Section V-C for upgrading HLA from "organization" to "Grid".

Granowetter [24] identifies the interoperability issue of different RTI implementations, and compares three possible approaches for resolving the issues, but these are still within in the HLA box as shown in Figure 3.

As a separate note, many papers discuss Web-based simulation, yet none has identified the importance of making interoperability practical. In fact, whether Web-based simulation is a revolution or evolution is discussed in [31], but the practical aspect of how interoperability is effected in Web-based simulation is not discussed either. We believe practical interoperability forms the basis of all cross-domain systems including simulation systems, and the related issues discussed in Section V-C are very critical for the success of distributed simulation on the Grid.

VI. HLAGrid

A. Design Overview

Ever since the invention of the World-Wide-Web, people around the world have experienced a new way of sharing information, but how software components on the Internet should interoperate to carry out applications is still not known. To tackle this challenge, there have been many technologies proposed, such as CORBA [5], RMI [9], DCOM [7], etc. The emerging Grid computing shares the vision of having the general public, administrator, decision maker, and organizations agree on open standards to form Virtual Organizations. Organizations, countries or even unions of countries (e.g. European Union) are deciding policies to deploy their Grid systems to interoperate with each other.

To extend HLA's principle of interoperability and reusability to the Grid, we leverage on the Grid infrastructure to extend HLA's practical interoperability to a new level. Our proposed layered architecture is shown in Figure 5. The ideal framework for distributed simulation sits on top of our HLAGrid and various other Grid projects, such as Access Grid [4], Semantic Grid [11], Data Grid [18], etc. A complete design will involve several aspects of practical interoperability in general, such as psychological, economic, legal, and philosophical aspects, which are out of the scope of this paper. We focus on the information technology aspect, which aims to enable distributed simulation execution using HLA on the Grid, and our goals include:

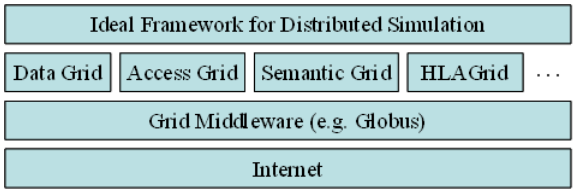

Fig. 5. Proposed layered architecture for distributed simulation on Grid.

- to provide a standard HLA API, for reasons of interoperability and reusability.
- to overcome the limitation of firewalls in traditional HLA/RTI implementations.
- to support migration of federates, or even the entire framework.
- to facilitate simulation model composition through Grid service composition.

The HLAGrid framework includes a Federate-Proxy-RTI architecture, in which different participants (clients) in the same simulation run their federate codes at their local sites, and the RtiExec and FedExec are executed on the remote resource. A new entity, *proxy*, is introduced to act on behalf of the clients' federate code to communicate with proxies of other clients through the RTI. Proxies are executed at remote grid resources. Federate codes and their respective proxies communicate with each other through Grid services, and a Grid-enabled HLA API, which provides the standard HLA API to the federate codes, is implemented to translate the communications into Grid services invocations. Our framework also includes additional Grid services to support the creation of the RTI, discovery of federations, etc.

In this framework, clients can run their simulator anywhere on any type of machine architecture. This framework hides the communication over the Grid network, and provides user transparency and simulator reusability. It also facilitates migration of federates without affecting other parts of the simulation, and the Proxy-RTI backbone can also be migrated, as it does not involve any simulation logic. Different RTI services can be exposed as separate Grid services, which provide standard state management schemes that are required for service composition. This framework incurs additional communication cost and overhead in embedding HLA communication into Grid service invocations, which will be quantified in the experiments of our implementation.

B. Detailed Design

The framework consists of two major components: client and resource (proxy-RTI backbone) together with supporting Grid services, which are interconnected through the Grid network, as shown in Figure 6.

1) Client Side: The client side provides the standard HLA API to the federate code, while allowing communication through the Grid. We make use of the Globus Grid middleware as the lower level communication channel, and provide a Grid-enabled API to translate federate-RTI communication into Grid service invocations, as illustrated in Figure 6.

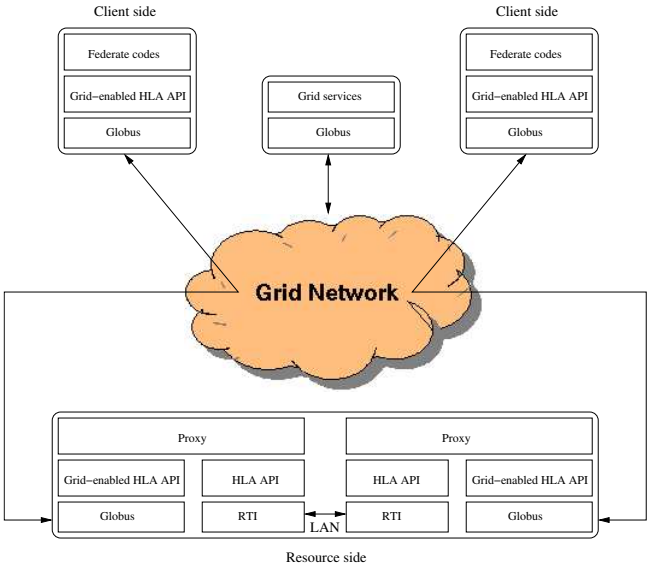

Fig. 6. Architecture of Proxy-based HLA simulation on the Grid.

To translate federate-RTI communication, all the method calls in the *RTIambassador* class and the callback functions provided in the *FederateAmbassador* need to be exposed to Globus. The *RTIambassador* method calls across the Grid are achieved using remote Grid service invocations by embedding the parameters inside invocations. When there is a *FederateAmbassador* callback from the RTI, the proxy will deliver the callback through Grid service invocation to the client. Figure 7 provides the conceptual view of the interactions between client side and the proxy.

Note that, the proxy decouples the client and RTI, and the simulation logic is maintained securely at the client side. The Proxy-RTI only provides mechanisms for simulation management, such as federation management, time management, etc. This architecture facilitates migration of the federate without affecting other parts of the simulation, because during the migration of the federate, the proxy could still respond to the RTI. There is no need for other federates to suspend their executions, which is required in [38]. Moreover, the proposed framework also supports migration of the Proxy-RTI backbone, because in the proposed framework, the Proxy-RTI backbone does not involve any simulation logic, but more things need to be done to support such features.

2) Resource Side: At the resource side, the *proxy* acts on behalf of the client and communicates with the RTI through the Local Area Network (LAN), as shown in Figure 8. The proxy is responsible for translating *RTIambassador* Grid service invocations into normal federate initiated RTI services, as well as embedding *FederateAmbassador* callbacks into Grid service invocation to the client's *FebAmb* Grid service.

3) Other Grid Services: Besides the Grid services to enable the communication between the client's federate code and the remote RTI, other Grid services are required for creating the RTI, discovering the federation, etc.

- RTI services: a persistent RTI service factory is needed to create instances of RTI services.
- Indexing services: a persistent indexing service is needed for maintaining the mapping between federations and handles of corresponding RTI services instances.

For how these services are created, managed and coordinated to provide various services for distributed simulation on Grid, please refer to [39] for more details.

C. HLA Simulation on the Grid Walk-through

We describe the detailed steps involved in a HLA simulation on the Grid.

- Startup stage: steps 1 to 5
  1) Create RTI: the federate code will invoke the persistent RTI service factory to create a RTI service instance, and use the instance to start the RTI at the resource side. The newly created RTI instance and the federation name need to be registered with the index service, so that other federates in the same federation will be able to look up the correct RTI instance. Note that, there is a concurrency issue involved in creating the RTI and registering the RTI with the indexing service. It can be resolved using a reservation table and a reference table, and the details are in [39].
  2) Create federate ambassador and RTI ambassador: the *RTIambassador* and *FederateAmbassador* will be created at the resource side, so that they will communicate with the RTI locally.
  3) Create and join federation execution: two *RTIambassador* method calls will be made at the client side and translated into Grid service invocations to the resource side.
  4) Initialize, publish and subscribe: all simulation settings will be initialized by invoking the *RTIambassador* at the resource side.
  5) Enable time constrained and time regulating if required as in step 4.
- Main loop: step 6
  6) This is the main body of the federate code, which includes *RTIambassador* method calls and *FederateAmbassador* callbacks.
- Shutdown: steps 7 to 9
  7) Resign from federation: this involves a method call to the remote *RTIambassador*.
  8) Destroy federation execution: besides the Grid service invocation to destroy the federation, the federation should be deregistered in the index service.
  9) Destroy RTI: this will be done according to the administrative rules at the resource side.

D. Preliminary Experiments and Results

In order to investigate the overhead incurred in the proposed framework, we converted the benchmark programs from DMSO's HLA package into Java programs, and tested them under different network configurations. We focused on two main benchmarks, i.e. Latency and Time Advancement.

The latency benchmark program measures RTI performance in terms of the latency of federate communications. More

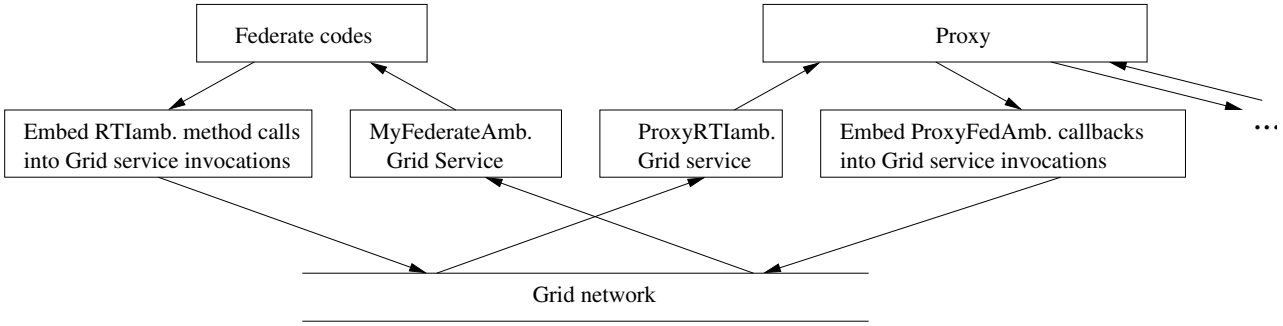

Fig. 7. Interaction between client side and the proxy in the framework.

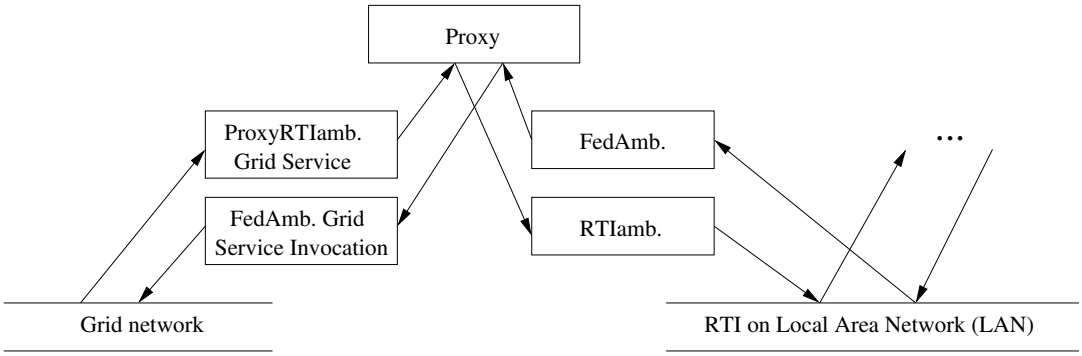

Fig. 8. Interaction between the proxy and the RTI at the resource side.

specifically, the benchmark program measures the elapsed time it takes for federates to send and receive an attribute update. The benchmark uses two federates, and it works as follows: one federate sends an attribute update, and upon receiving this update, the other federate sends it back to the sending federate. The elapsed time of this communication is calculated by using the timestamps taken at the sending and reflecting federates. All parameters used are the default value given in the DMSO package. There is no queuing of messages involved, and the communication payload is assumed to be negligible. The communication protocol used is reliable.

The time advancement benchmark program measures RTI performance in terms of the rate at which time advance requests are processed. The benchmark uses two federates with timestep cycle of 10, all other parameters are the default values from the DMSO package.

The testing is done using the Linux cluster in the Parallel and Distributed Computing Center in the School of Computer Engineering of Nanyang Technological University in Singapore and a Linux workstation in the School of Computer Science in Birmingham University in the United Kingdom.

There are five components in the experiment: RTI, two federates $Federate1$ and $Federate2$, two more corresponding proxies $Proxy1$ and $Proxy2$ required by the HLAGrid software. The processes $RtiExec$ and $FedExec$ are executed in Singapore on machines M_{rti}. The processes $Federate1$, $Proxy1$, and $Proxy2$ are executed in Singapore on machines M_{fed1}, M_{proxy1}, and M_{proxy2} respectively. The process $Federate2$ is executed on M_{fed2}-SG in NTU, and on M_{fed2}-UK in Birmingham as a comparison. All machines in NTU are inter-connected using Myrinet with connection speed of 1Gbit per second. There is a connection between machine M_{proxy2} in NTU to machine M_{fed2}-UK in Birmingham through the Grid network. The experiment hardware configuration is shown in Figure 9. Individual machines' specifications are shown in Table I. Figure 10 shows the configuration of the latency benchmark and the major communication involved. As a comparison study, the same benchmark programs are executed in the cluster and Wide-Area-Network (WAN) using both the DMSO HLA's implementation and our HLAGrid prototype.

The experimental results are shown in Table II. The cluster version of the latency benchmark shows that our prototype incurs about 40 millisecond of overhead, and this is mainly due to the use of Globus, and encoding/decoding of parameters/result. The latency of the WAN version is much larger than the cluster version, and the latency in the HLAGrid is about 3-4 times that of the HLA. We use the Unix command TCPDUMP to monitor the traffic (SOAP messages) involved in Grid service requests/responses. We observe that the size of the SOAP message is around 1-2 KByte per Grid service request/response, which is much larger than the attribute size used in the benchmark (128 byte), and the overhead involved are the namespace information and XML tags. Considering the additional marshaling and unmarshaling of the SOAP messages required in a Grid service invocation, the Grid service communication overhead is much larger than the pure socket connection in the existing HLA. The time-advancement benchmark also shows similar results.

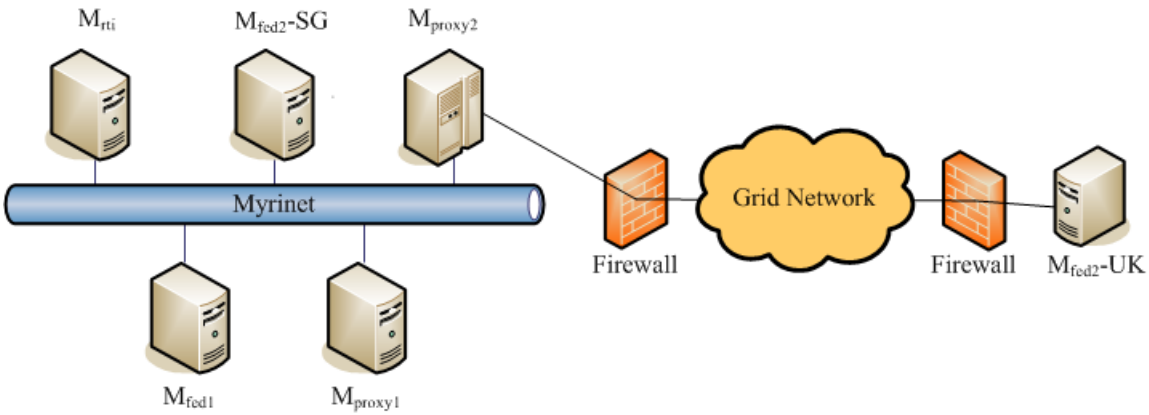

Fig. 9. Experiment hardware configuration.

TABLE I
SPECIFICATION OF MACHINES FOR EXPERIMENTS.

	M_{proxy2}	M_{rti}, M_{fed1}, M_{fed2}-SG	M_{proxy1}	M_{fed2}-UK
CPU	4xPentiumIII 500MHz	PentiumIII 733MHz	2xPentiumIII 733MHz	AMD Athlon 1.5GHz
Memory	1 Gbyte	1 Gbyte	1 Gbyte	2 Gbyte
OS	Redhat Linux 7.0	Redhat Linux 7.0	Redhat Linux 7.0	Redhat Linux 7.3
gcc	3.0.2	3.0.2	3.0.2	3.0.2
HLA	DMSO NG 1.3 V6	DMSO NG 1.3 V6	DMSO NG 1.3 V6	DMSO NG 1.3 V6

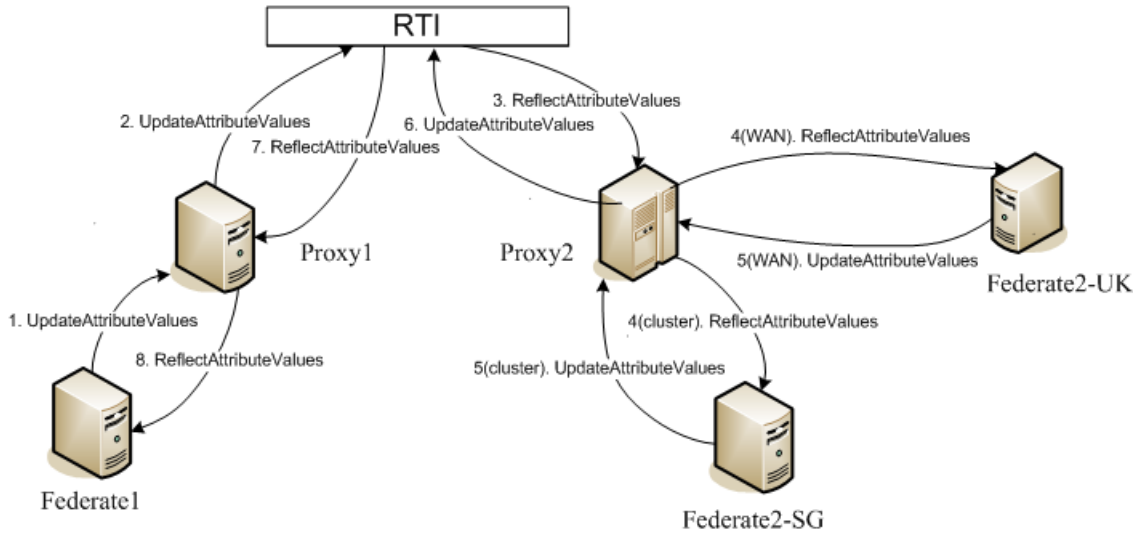

Fig. 10. Latency benchmark configuration.

TABLE II
EXPERIMENT RESULTS

		HLA	HLAGrid
Latency	Cluster	10 millisecond	50 millisecond
	WAN	305 millisecond	1200 millisecond
Time Advancement	Cluster	680 grants/second	150 grants/second
	WAN	2 grants/second	0.41 grants/second

VII. Conclusions and Future Work

In this paper, we focus on two important aspect of modeling and simulation: *interoperability* and *composability*. We propose an interoperability model based on both conceptual and practical aspects and extend the existing HLA's practical interoperability to support Grid-wide distributed simulation. The HLAGrid framework achieves interoperability between different simulators (federates) by using a Federate-Proxy-RTI architecture. This architecture hides the heterogeneity of the simulators, simulators' execution platforms, and how the simulators communicate with the RTI. Moreover, RTI services are exposed as Grid services, which provides more secure, scalable and coordinated management. RTI services' internal data are exposed as Grid service data elements, which allows both *pull* and *push* kind of access by other Grid services. All interfaces used in the framework comply with the standard HLA interface specification, which provides reusability to simulators. A prototype of the framework is implemented using DMSO's RTI 1.3NG version 6 and the Grid system runs the Globus Toolkit Version 3 to achieve compatibility and interoperability. Experimental results show that the our prototype incurs more overhead than the existing HLA, and is suitable for coarse-grained applications. Much work remains to be done for model execution, such as federate migration, fault-tolerance, integration with security, etc. To support model composition, a framework that addresses model development, sharing, and validation is being developed.

References

[1] "BEEP: Blocks extensible exchange protocol," 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3080.txt

[2] "SOAP: Simple object access protocol," 2003. [Online]. Available: http://www.w3.org/TR/soap/

[3] "U.S. department of defense, chief information officer (cio), department of defense net-centric data strategy," The Pentagon, Washington, Tech. Rep., 2003.

[4] "Access grid (ag) project," 2004. [Online]. Available: http://www.accessgrid.org/

[5] "Common object request broker architecture (corba)," 2004. [Online]. Available: http://www.omg.org/gettingstarted/corbafaq.htm

[6] "Cross grid," 2004. [Online]. Available: http://www.eu-crossgrid.org/

[7] "Distributed component object model (dcom)," 2004. [Online]. Available: http://www.microsoft.com/com/default.mspx

[8] "Extensible modeling and simulation framework (xmsf)," 2004. [Online]. Available: http://www.movesinstitute.org/xmsf/xmsf.html

[9] "Java remote method invocation (java rmi)," 2004. [Online]. Available: http://java.sun.com/products/jdk/rmi/

[10] "Open gis consortium (ogc)," 2004. [Online]. Available: http://www.gis.com/software/ogc.html

[11] "Semantic grid project," 2004. [Online]. Available: http://www.semanticgrid.org/

[12] "Semantic web," 2004. [Online]. Available: http://www.w3.org/2001/sw/

[13] "Uk e–science project," 2004. [Online]. Available: http://www.rcuk.ac.uk/escience/

[14] T. Andrews, "Business process execution language," 2001. [Online]. Available: http://www-106.ibm.com/developerworks/library/ws-bpel/

[15] D. Berardi, D. Calvanese, D. Giacomo, and M. Mecella, "Reasoning about actions for e-service composition," in *ICAPS 2003 workshop on planning for web services*, 2003.

[16] J. S. Dahmann, F. Kuhl, and R. Weatherly, "Standards for simulation: as simple as possible but not simpler the high level architecture for simulation," *Simulation*, vol. 71, no. 6, pp. 378–387, June 1998.

[17] P. Davis and R. Anderson, *Improving the Composability of Department of Defense Models and Simulations*. RAND Corporation, 2003.

[18] G. D. G. Effort, "Globus data grid effort," 2004. [Online]. Available: http://www.globus.org/datagrid/

[19] J. B. Fitzgibbons, R. Fujimoto, D. Fellig, D. Kleban, and A. J. Scholand, "Idsim: an extensible framework for interoperability distributed simulation," in *Proceedings of the IEEE International Conference on Web Services (ICWS2004)*, 2004, pp. 532–539.

[20] I. Foster, "The anatomy of the Grid: Enabling scalable virtual organizations," *Lecture Notes in Computer Science*, vol. 2150, pp. 1–12, 2001. [Online]. Available: citeseer.ist.psu.edu/foster01anatomy.html

[21] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, no. 2, pp. 115–128, Summer 1997. [Online]. Available: citeseer.ist.psu.edu/article/foster96globus.html

[22] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The physiology of the grid: An open grid services architecture for distributed systems integration," 2002. [Online]. Available: www.globus.org\/research\/papers\/ogsa.pdf

[23] "Globus toolkit version 3." [Online]. Available: http://www.globus.org/

[24] L. Granowetter, "Rti interoperability issues – api standards, wire standards, and rti bridges," in *Proceedings of the 2003 European Simulation Interoperability Workshop*, no. 03S-SIW-063, 2003.

[25] R. Grønmo, D. Skogan, I. Solheim, and J. Oldevik, "Model-driven web services development," *International Journal of Web Services*, 2004.

[26] F. Leymann, ""web service flow language (wsfl 1.0)"," 2001. [Online]. Available: http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[27] Q. Liang, L. Chakarapani, S. Su, R. Chikkamagalur, and H. Lam, "A semi-automatic approach to composite web service discovery, description and invocation," *International Journal of Web Services*, 2004.

[28] S. McIlraith and C. Son, "Adapting golog for composition of semantic web services," in *8th conference on knowledge representation and reasoning*, 2002.

[29] D. M. Moen and J. M. Pullen, "Enabling real-time distributed virtual simulation over the internet using host-based overlay multicast," in *Proceedings of the IEEE/ACM Distributed Simulation-Real Time Application Symposium*, 2003, pp. 30–36.

[30] K. L. Morse, D. L. Drake, and R. P. Brunton, "Web enabling an *RTI* - an *XMSF* profile," in *Proceedings of the IEEE 2003 European Simulation Interoperability Workshop*, no. 03E-SIW-046, 2003.

[31] E. H. Page, A. Buss, P. A. Fishwick, K. J. Healy, R. E. Nance, and R. J. Paul, "Web-based simulation: revolution or evolution," *ACM Transaction on Modeling and Computer Simulation*, vol. 10, no. 1, pp. 3–17, January 2000.

[32] R. Soley and the OMG Staff Strategy Group, "Model-driven architecture," Object Management Group (OMG), Tech. Rep., November 2000.

[33] S. Thakkar, A. Knoblock, and L. Ambite, "A view integration approach to dynamic composition of web services," in *ICAPS 2003 workshop on planning for web services*, 2003.

[34] A. Tolk, "Avoiding another green elephant - a proposal for the next generation hla based on model driven architecture," in *Proceedings of the 2002 Fall Simulation Interoperability Workshop*, no. 02F-SIW-004, 2002.

[35] A. Tolk and J. A. Muguira, "The levels of conceptual interoperability model," in *Proceedings of the 2003 Fall Simulation Interoperability Workshop*, no. 03F-SIW-007, 2003.

[36] S. Tuecke, K. Czajkowski, I. Foster, J. Rey, F. Steve, and G. Carl, "Grid service specification," 2002. [Online]. Available: www.gridforum.org/ogsi-wg/drafts/GS_Spec_draft03_2002--07--17.pdf

[37] A. Wytzisk, I. Simonis, and U. Raape, "Integration of hla simulation models into a standized web service world," in *Proceedings of the 2003 European Simulation Interoperability Workshop*, no. 03E-SIW-019, 2003.

[38] K. Zajac, M. Bubak, M. Malawski, and P. M. A. Sloot, "Towards a grid management system for *HLA*-based interactive simulations," in *Proceedings of Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2003)*, S. Turner and S. Taylor, Eds. Delft, The Netherlands: IEEE Computer Society, October 2003, pp. 4–11.

[39] W. Zong, Y. Wang, W. Cai, and S. J. Turner, "Grid services and service discovery for hla-based distributed simulation," in *Proceedings of the IEEE/ACM Distributed Simulation-Real Time Application Symposium*, 2004.