Y. Xie, Y.M. Teo, W. Cai and S.J. Turner, Service Provisioning for HLA-based Distributed Simulation on the Grid, Proceedings of the 19th ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation, pp. 282-291, IEEE Computer Society Press, Monterey, USA, June 2005.

Service Provisioning for HLA-based Distributed Simulation on the Grid

Yong Xie[1], Yong Meng Teo[1,2], Wentong Cai[3], Stephen John Turner[3]
[1]Singapore-Massachusetts Institute of Technology Alliance
[2]Department of Computer Science, National University of Singapore, Singapore 117543
[3]School of Computer Engineering, Nanyang Technological University, Singapore 639798
smaxy@nus.edu.sg, teoym@comp.nus.edu.sg, {aswtcai, assjturner}@ntu.edu.sg

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 provisioning of the services is becoming increasingly difficult. The Grid provides a platform for coordinated resource sharing and application development and execution. In this paper, we focus on extending the High Level Architecture (HLA) to the Grid. More specifically, we propose a distributed simulation framework, called HLAGrid. The framework uses a Federate-Proxy-RTI architecture, which allows resources on the Grid to be utilized on demand by using Grid services. The architecture also supports federation discovery, security of the simulator logic, and flexible federation construction, such as hierarchical federations. The architecture hides the heterogeneity of the simulators, simulators' execution platforms, and how the simulators communicate with the RTI. 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.

1. 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 [5], 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 [5].

In order to run a distributed simulation over the Wide-Area-Network (WAN) using the IEEE HLA/RTI directly, special arrangements have to be made beforehand to ensure the availabilities of the required hardware and software. Such arrangements are typically made with a centralized control or simply within an organization, because inter-organizational sharing of resources involves issues such as security. With the increase in the scale and complexity of simulations, large amounts of resources are required, and provisioning of services becomes more and more difficult. In the case of HLA-based distributed simulation, the resources of RTI, the simulation model, and/or the underlying infrastructure may not be available when they are needed.

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 [9]. Much effort has been made in the area of Grid computing since 2001. Many systems and middleware for Grid computing have been proposed in the past few years, and Globus [10] is becoming the de facto standard middleware for Grid computing. The third version of the Globus Toolkit [12] is based on the concept of *Grid Services*, which is defined by the Open Grid Services Architecture (OGSA) [11], and is specified by the Open Grid Services Infrastructure (OGSI) [21]. The Grid Services architecture enables resources to be dynamically discovered and shared on demand, thus supports the provisioning of services for large-scale Grid applications, such as HLA-based distributed simulation.

In this paper, we aim to extend the IEEE HLA to the Grid, and more specifically we focus on the provisioning of services to support HLA-based distributed simulation on the Grid. We present a distributed simulation framework to address several important issues, such as HLA/RTI service provisioning, heterogeneity, federation discovery, federation construction, and security. The framework supports the provisioning of the RTI service 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, RTI services are exposed as Grid services, and the communications are through Grid service invocation, which provides more secure, scalable and coordinated management. The architecture also supports dynamic discovery of the federation, and hierarchical federations. 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.

The rest of the paper is organized as follows: Section 2 presents the literature review on related work. Section 3 discusses the design of the Federate-Proxy-RTI architecture and our prototype implementation. In Section 4, we describe the experimental results. Section 5 presents further discussion on the merits of our proposed architecture. Our concluding remarks and plan for future work are in Section 6.

2. Literature Review

In the literature on improving the HLA, some researchers [13, 22] focus on building tools to ease the process of modeling and simulation using HLA, and some [24] focus on adding auxiliary systems to make HLA more useful in a wide-area-network (WAN), while others [7, 8, 19, 20] have a more ambitious objective to reinvent HLA to be model-driven and composable, or replace HLA with a new framework.

Some researchers focus on the extension of HLA's interoperability. For example, in [19, 20], Tolk, et al. propose a framework to integrate HLA into the Model-Driven Architecture (MDA) [18] defined by the Object Management Group (OMG) to improve interoperability.

Wytzisk, et al. [22] propose a solution that brings HLA and the Open GIS Consortium (OGC) [15] 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. [24] propose a system to enable HLA-based simulations on the Grid, but they focus on migrating federates. The system also includes discovery, and information indexing services. However, the communication between RTI and federates is based on the original HLA communication, which requires predefined ports to be open. Fitzgibbons, et al. [8] present a distributed simulation framework, called IDSim, based upon OGSI [21]. 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.

The distributed simulation communities initiated a plan, called the Extensible Modeling and Simulation Framework (XMSF) [7], 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. [14] 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) [17] and employing the Blocks Extensible Exchange Protocol (BEEP) [2] communication layer to enable bi-directional calls and callbacks via web services.

Compared with all the previous work, we have different focuses, which are the provisioning of services and flexible construction of federations. We believe these are the characteristics that will differentiate a Grid-enabled HLA-based distributed simulation from the traditional IEEE HLA/RTI based distributed simulation.

3. Design and Implementation

3.1. Overview

To provide services to support HLA-based distributed simulation on the Grid, we leverage on the existing Grid infrastructure, and the layered framework is shown in Figure 1. The ideal framework for distributed simulation over the Grid should sit on top of our HLAGrid and various other Grid projects, such as Access Grid [1], Semantic Grid [16], and Data Grid [6]. Our specific goals include:

- to support RTI service to be used on demand
- to support dynamic discovery of federations
- 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 hierarchical federations

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

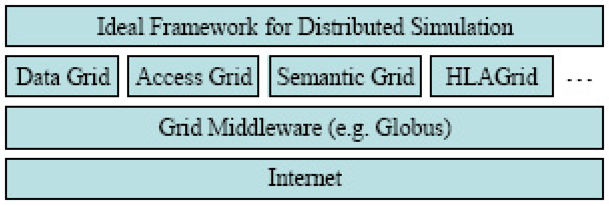

Ideal Framework for Distributed Simulation
Data Grid
Access Grid
Semantic Grid
HLAGrid
...
Grid Middleware (e.g. Globus)
Internet

Figure 1. A layered framework for distributed simulation on the Grid.

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 and discovery of federations. As a result, the RTI resource can be created, discovered, and used on demand. Moreover, hierarchical federations can be constructed by using the proxy in a way similar to the mechanisms used in [3, 4]. Users have the flexibility to join the federation locally, remotely (using the Federate-Proxy-RTI architecture), or hierarchically.

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 Grid 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.

3.2. 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 2.

3.2.1. 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 2.

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 3 provides the conceptual view of the interactions between the client side and the proxy.

Note that, the proxy decouples the client and the 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 and time management. 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 [24]. 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.

3.2.2. 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 4. 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 invocations to the client's *FebAmb* Grid service.

3.2.3. 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 and discovering the federation.

- RTI services: persistent RTI service factories are needed to create instances of RTI services.
- Indexing service: 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 the Grid, please refer to [25] for more details.

3.3. HLA Simulation on the Grid Walk-through

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

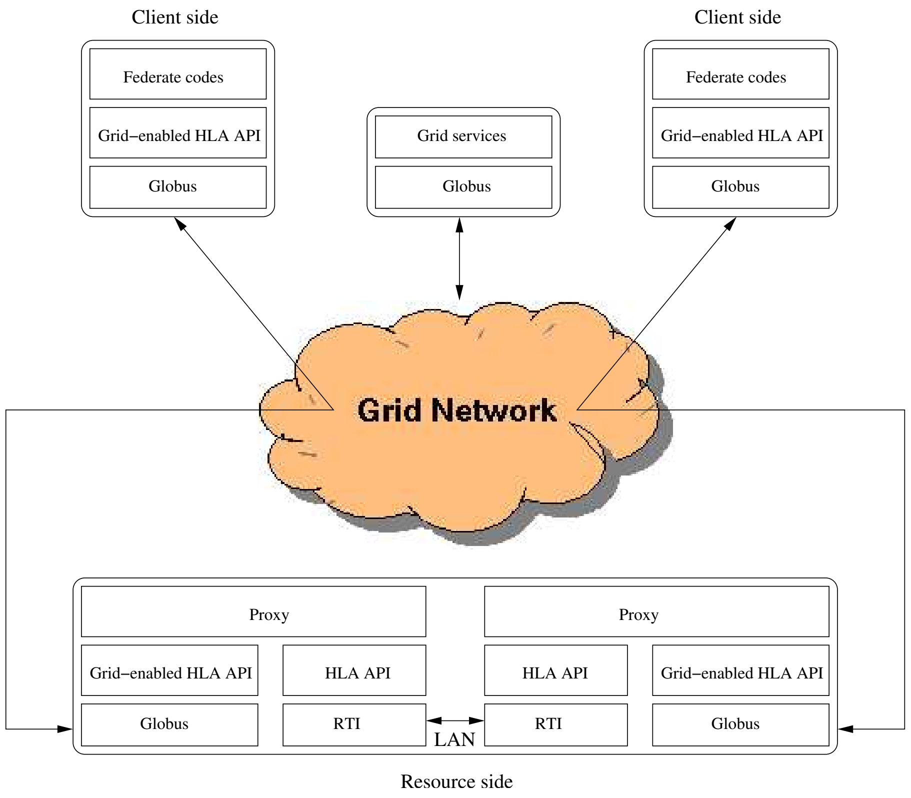


Figure 2. Architecture of Proxy-based HLA simulation on the Grid.

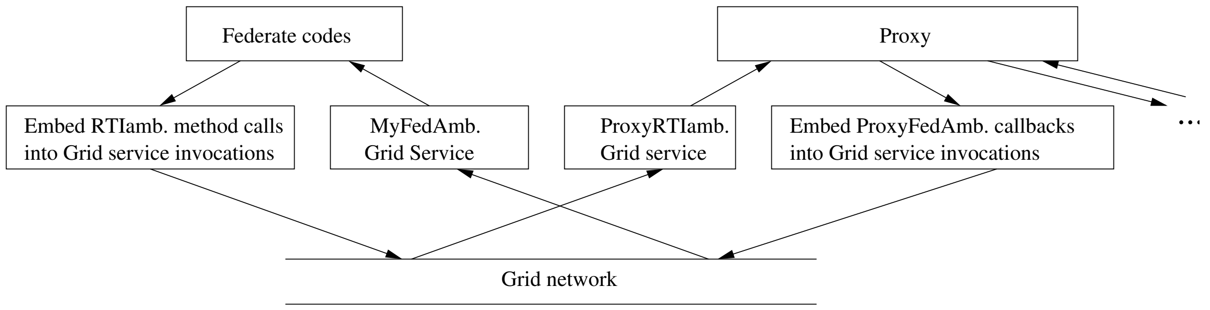

Federate codes
Proxy
Embed RTIamb. method calls into Grid service invocations
MyFedAmb. Grid Service
ProxyRTIamb. Grid service
Embed ProxyFedAmb. callbacks into Grid service invocations
...
Grid network

Figure 3. Interaction between client side and the proxy in the framework.

- 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 run *rtiexec* 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 index service. It can be resolved using a reservation table and a reference table, and the details are in [25].

  2. Create federate ambassador and RTI ambassador: the *Proxy* is created, and its associated *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 *RTI-ambassador* method calls will be made at the client side and translated into Grid service invo-

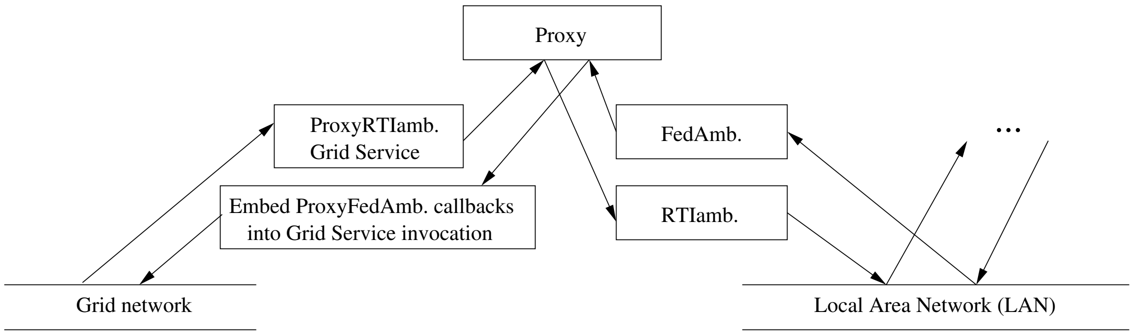

Proxy
ProxyRTIamb. Grid Service
FedAmb.
...
Embed ProxyFedAmb. callbacks into Grid Service invocation
RTIamb.
Grid network
Local Area Network (LAN)

Figure 4. Interaction between the proxy and the RTI at the resource side.

cations to the resource side.

4. Initialize, publish and subscribe: all simulation settings will be initialized by invoking the *RTI-ambassador* 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, as shown in Figure 5 and Figure 6.

- 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.

3.4. Implementation

A prototype of our proposed framework is implemented in Java, and the Grid system runs the Globus Toolkit version 3.

At the client side, three main classes are implemented, as shown in Figure 7:

- *HLAGridNullFedAmb* class extends the default *NullFederateAmbassador* class provided in the DMSO's HLA implementation, and all the interfaces remain the same.
- *ClientProvider* class is a Grid service. An instance of the *ClientProvider* class will be able to receive Grid service invocation from the Proxy and translate them into callbacks of *MyFedAmb*.

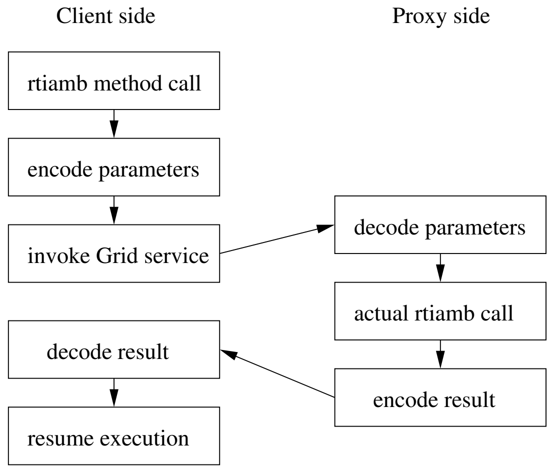


Figure 5. Executing a *RTIambassador* method call through Grid service invocation.

- *HLAGridRTIAmb* embeds all the *RTIambassador* calls into a Grid Service invocation.

The client's own code should contain an instance of *HLAGridRTIAmb* and an instance of the client-defined *MyFedAmb* which extends the *HLAGridNullFedAmb* class.

At the resource side, five main classes are used:

- *ProxyServiceProvider* class implements the Grid service class *OperationProvider*. It accepts remote invocations of *RTIambassador* method calls, and delivers *FederateAmbassador* callbacks to the client by invoking the Client's Grid service.
- *ProxyNullFedAmbassador* extends the default *NullFedAmbassador* in the DMSO's HLA implementation.
- *ProxyRTIComponent* contains a *RTIambassador* to communicate with the RTI.

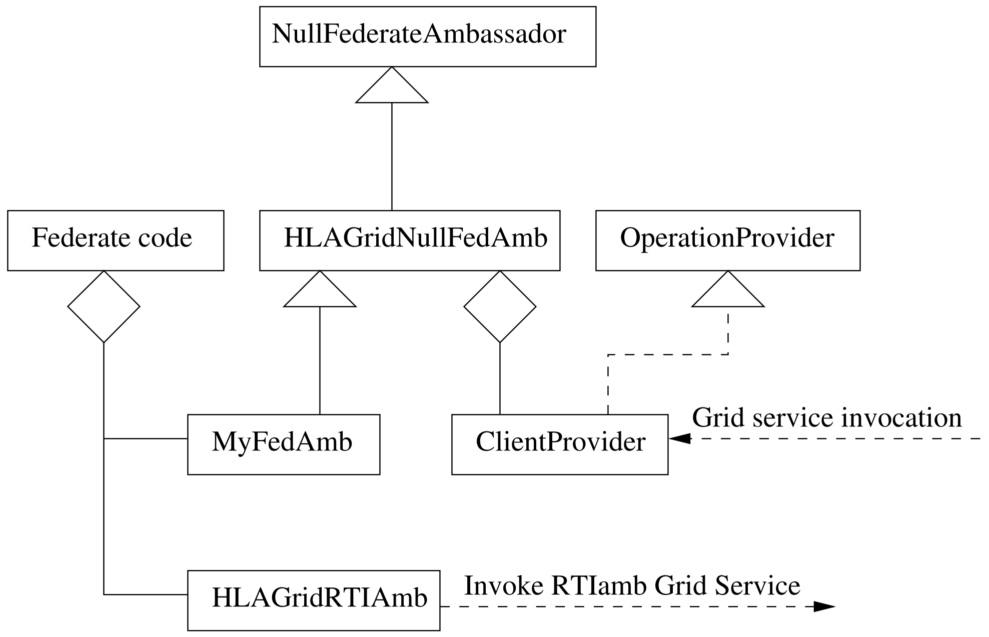


Figure 7. The class diagram of the client side.

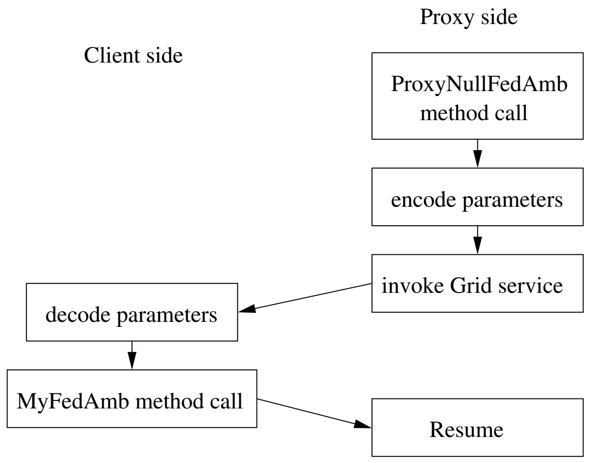


Figure 6. Executing a *FederateAmbassador* callback through Grid service invocation.

- *ProxyFedComponent* also contains a *ProxyNullFedAmbassador*, which embeds callbacks into Grid Service invocations.

We have implemented the *RTIambassador* services' API for Federation Management, Time Management, Object Management, Declaration Management, and Ownership Management. As mentioned in [14], to implement the Data Distribution Management (DDM), we need to keep the state and context for each region and its dimension handles. We plan to implement the DDM in future. A detailed description of the implementation can be found in [23].

4. 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 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 real time taken at the sending and reflecting federates. All parameters used are the default values 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.

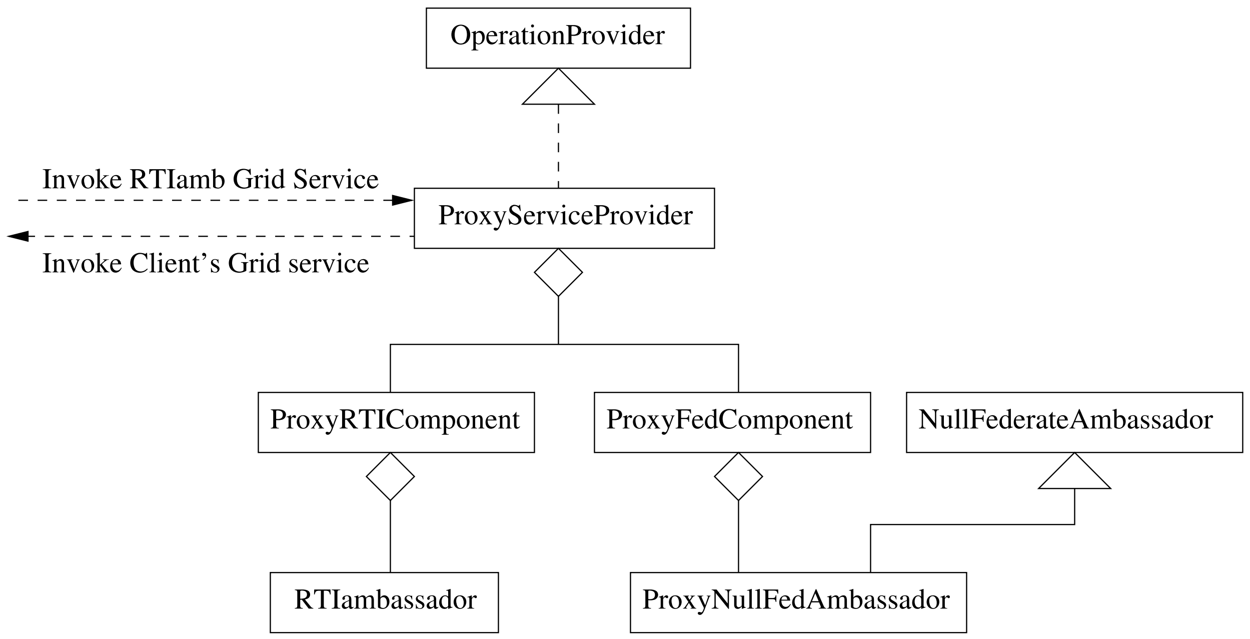


Figure 8. The class diagram of the resource side.

The testing is done using the Linux cluster in the Parallel and Distributed Computing Centre at the School of Computer Engineering of Nanyang Technological University in Singapore and a Linux workstation in the School of Computer Science at 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 ethernet. 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 1. 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. Besides the configuration shown in Table 1 for benchmark experiments, we have also demonstrated our HLAGrid on more heterogenous platforms between Singapore and UK Midland e-Science center at Birmingham University. For example, in the e-Science cluster at Birmingham University, each node has 2 GBytes of memory and 2 Intel Xeon 3GHz processors, with Red Hat Enterprise Linux (RHEL) system version AS 3 as the operating system.

The experimental results are shown in Table 2. The cluster version of the latency benchmark shows that our prototype incurs about 40 millisecond of overhead in a cluster, and this is mainly due to the use of Globus, the encoding/decoding of parameters/result, and the communication cost. The overhead of Globus and the encoding/decoding of parameters/result are relatively fixed. But the communication cost varies significantly between the WAN version and the cluster version. From the comparison between HLA's cluster and WAN version, we can calculate the difference in communication cost is roughly 300 millisecond. Also, we can observe that in the HLAGrid's cluster and WAN latency benchmark comparison, the difference in communication cost is around 1150 milliseconds, which is about 3-4 times that of the HLA case. To understand this, 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 overheads involved are the namespace information and XML tags. The increase of the number of packets for the communication through Grid service invocation as compared to the pure socket connection in the existing HLA significantly affects the communication efficiency. The time-advancement benchmark also shows similar results.

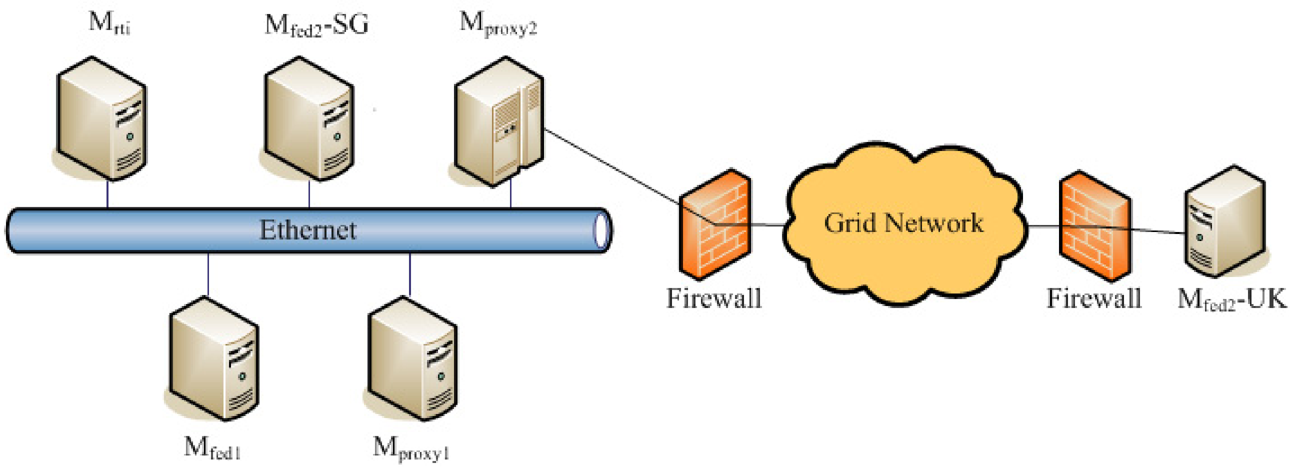
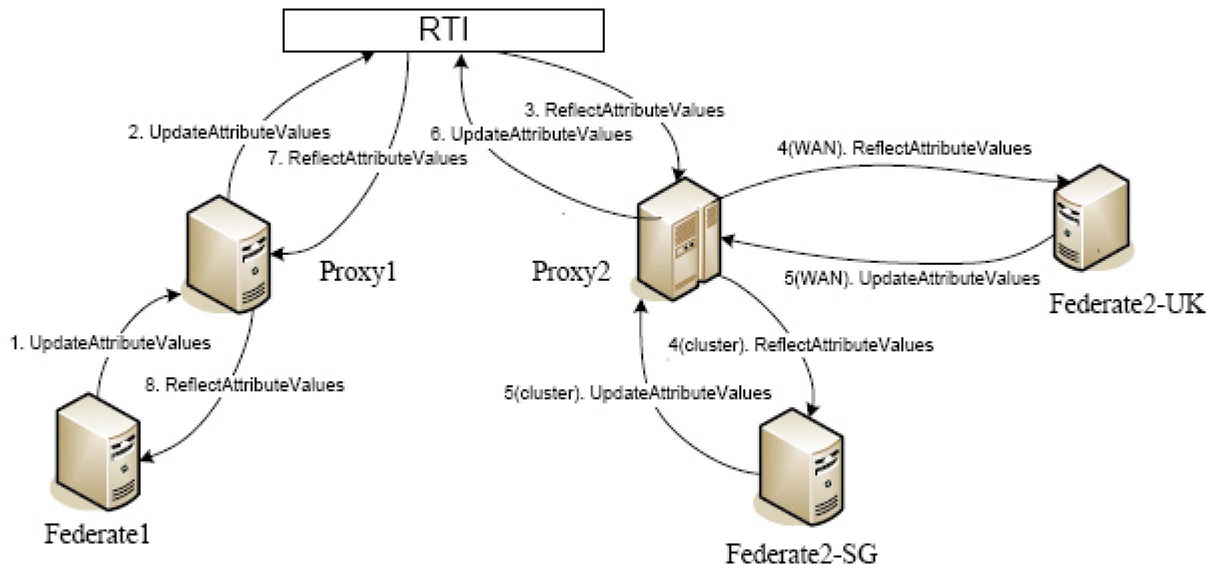


Figure 9. Experiment hardware configuration.

	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

Table 1. Specification of machines for experiments.



RTI
2. UpdateAttributeValues
7. ReflectAttributeValues
3. ReflectAttributeValues
6. UpdateAttributeValues
4(WAN). ReflectAttributeValues
5(WAN). UpdateAttributeValues
Proxy1
Proxy2
Federate2-UK
1. UpdateAttributeValues
8. ReflectAttributeValues
4(cluster). ReflectAttributeValues
5(cluster). UpdateAttributeValues
Federate1
Federate2-SG

Figure 10. Benchmark configuration.

		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

Table 2. Experiment results

5. Benefits of the Architecture

Despite the rather lackluster performance results, the proposed architecture supports service provisioning for HLA-based distributed simulation on the Grid in several aspects.

Firstly, clients only need the Grid-enabled HLA library for the federates to collaborate with other ones. This library differs from the HLA/RTI JavaBinding, because it does not require the HLA/RTI runtime executable code, while the HLA/RTI JavaBinding executes such code via Java Native Interface (JNI). Thus, the Grid-enabled HLA Library does not rely on any HLA/RTI implementation, providing more flexibility to the users. Moreover, the Grid-enabled HLA library is implemented using Java, and it hides the heterogeneity of the platforms (CPU, Operating System, etc) where federates can be executed.

Secondly, as one of the main features of the proposed framework, the actual HLA/RTI services are provided over the Grid, rather than provisioned by the clients who need to run the federates. From the architecture's point of view, this feature allows users to run simulations without having to manage the RTI service. Also, the entire simulation can be provided as a service, so that other clients can discover and use this service via the Grid-enabled HLA library. For example, if there exists a HLA-based supply chain management simulation available on the Grid, a wafer fab federate can make use of the proposed framework to get involved in the simulation without the knowledge of the HLA/RTI version and the details of how the supply chain management simulation is constructed, or how other federates are implemented. The proposed framework provides transparency not only between federates and the RTI, but also between a federate and the rest of the federation because of the provisioning of services (both RTI service and the simulation as a service).

Besides, multiple copies of the HLA/RTI services can serve multiple federations. The HLA/RTI service could be provided by different parties on heterogenous platforms, and this increases the availability of the HLA/RTI services. Clients can choose which HLA/RTI service to use, which provides more flexibility to the users. Furthermore, with the availability of multiple HLA/RTI services, load balancing among the service providers is possible. Even if one HLA/RTI service fails due to reasons like a hardware crash, other services can still be used by clients, providing fault-tolerance to the system as a whole.

Furthermore, clients can discover services and join a federation dynamically. In the traditional HLA-based distributed simulation, special arrangement has to be made beforehand. But, with the proposed architecture, the HLA/RTI service is discovered dynamically, and federates can join into an existing federation without prior arrangement. This feature is especially suitable for interactive simulation such as war-game simulation, where federates (aircrafts, tanks, troops, etc) join and leave the federation dynamically.

Lastly, the proxy and the Grid-enabled HLA library in the proposed architecture provides transparent HLA/RTI method calls and callbacks through Grid service invocations, and can be used as gateway/bridge to support hierarchical federations by connecting multiple RTIs [3, 4].

6. Conclusion and Future Work

In this paper, we propose a framework to extend the HLA to support Grid-wide distributed simulation. More specifically, we focus on provisioning of resources through Grid services and flexible construction of hierarchical federations. The framework achieves interoperability between different simulators (federates) by using a Federate-Proxy-RTI architecture. This architecture allows federates with heterogenous platforms to dynamically discover the RTI service and join a federation on the Grid. Moreover, the RTI services are exposed as Grid services, which provides more secure, scalable and coordinated management. 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 DMSO HLA/RTI, and is suitable for coarse-grained applications. Much work remains to be done, such as federate migration, fault-tolerance, and integration with security.

References

[1] Access Grid Project. http://www.accessgrid.org/, 2004.

[2] BEEP. Blocks extensible exchange protocol, http://www.ietf.org/rfc/rfc3080.txt, 2003.

[3] W. Cai, G. Li, S. J. Turner, B.-S. Lee, and L. Liu. Automatic construction of hierarchical federations architecture. In *Proceedings of Seventh IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2002)*, pages 50–58, 2002.

[4] W. Cai, S. J. Turner, and B.-P. Gan. Hierarchical federations: an architecture for information hiding. In *Proceedings of Fifteenth Workshop on Parallel and Distributed Simulation*, pages 67–74, 2001.

[5] 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*, 71(6):378–387, June 1998.

[6] Data Grid Project. http://www.globus.org/datagrid/, 2004.

[7] Extensible Modeling and Simulation Framework (XMSF). http://www.movesinstitute.org/xmsf/xmsf.html, 2004.

[8] J. B. Fitzgibbons, R. Fujimoto, D. Fellig, D. Kleban, and A. J. Scholand. IDSim: An extensible framework for interoperable distributed simulation. In *Proceedings of the IEEE International Conference on Web Services (ICWS2004)*, pages 532–539, 2004.

[9] I. Foster. The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, 2150:1–12, 2001.

[10] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

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

[12] Globus Toolkit version 3. http://www.globus.org/.

[13] L. Granowetter. RTI interoperability issues – API standards, wire standards, and RTI bridges. In *Proceedings of the 2003 European Simulation Interoperability Workshop*, number 03S-SIW-063, 2003.

[14] 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*, number 03E-SIW-046, 2003.

[15] Open GIS Consortium (OGC). http://www.gis.com/software/ogc.html, 2004.

[16] Semantic Grid Project. http://www.semanticgrid.org/, 2004.

[17] SOAP. Simple Object Access Protocol, http://www.w3.org/tr/soap/, 2003.

[18] R. Soley and the OMG Staff Strategy Group. Model-driven architecture. Technical report, Object Management Group (OMG), November 2000.

[19] 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*, number 02F-SIW-004, 2002.

[20] A. Tolk and J. A. Muguira. The levels of conceptual interoperability model. In *Proceedings of the 2003 Fall Simulation Interoperability Workshop*, number 03F-SIW-007, 2003.

[21] S. Tuecke, K. Czajkowski, I. Foster, J. Rey, F. Steve, and G. Carl. Grid service specification, www.globus.org/research/papers/gsspec.pdf, 2002.

[22] 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*, number 03E-SIW-019, 2003.

[23] Y. Xie, Y. M. Teo, W. Cai, and S. J. Turner. A distributed simulation backbone for executing HLA-based simulation over the internet. In *Workshop on Grid Computing & Applications, Proceedings of the 2nd International Conference on Scientific and Engineering Computation*, pages 96–103, June 2004.

[24] 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)*, pages 4–11, Delft, The Netherlands, October 2003.

[25] 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*, pages 116–124, 2004.