A Distributed Simulation Backbone for Executing HLA-based Simulation Over the Internet

Yong Xie[1], Yong Meng Teo[1,2], Wentong Cai[3] and Stephen John Turner[3]
[1]Singapore-Massachusetts Institute of Technology Alliance, Singapore 117576
[2]Department of Computer Science, National University of Singapore, Singapore 117543
[3]School of Computer Engineering, Nanyang Technological University, Singapore 639798
Email: xieyong@mit.edu

Abstract

With the increase in complexity of simulations, there is a growing need for accessing large amount of computing resource and data sets which could be geographically distributed. The Grid provides an infrastructure for collaborative resource sharing to execute large-scale applications. In this paper, we propose a framework for distributed simulation using the High Level Architecture (HLA) over the Grid. The backbone makes use of a Federate-Proxy-RTI architecture, in which a remote proxy acts on behalf of the federate in interacting with RTI. This architecture hides the communication over the Grid network, and provides user transparency and simulator reusability. A prototype is implemented using DMSO's RTI 1.3 and the Grid system runs the Globus Toolkit Version 3 to achieve compatibility and interoperability.

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 [3], and it is 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 conforming to the standard HLA specification [3].

Complex simulations often require large amount of computing resource and the data sets required could be geographically distributed. To cater for the growing complexity of the simulation systems, it is necessary to harness the computing power over the internet. The concept of "Grid" computing was proposed by Ian Foster as a coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [4]. Many systems for Grid computing have been proposed in the past few years, and Globus [5] is becoming the de facto standard middleware for Grid computing. The third and latest version of the Globus Tookit [10] is based on the concept of *Grid Services*, which is defined by the Open Grid Service Architecture (OGSA) [6], and is specified by Open Grid Service Infrastructure (OGSI) [9].

In order to cater for the increasing complexity of distributed simulation systems, it is necessary to harness the computing resources on the Internet. Grid technology enables internet-wide sharing of computing resource, and also facilitates the access of data sets across the internet. However, the current industrial standard for distributed simulation, HLA, is typically used in a cluster or a Local Area Network (LAN) with single administrative domain. It is difficult to use HLA on wide area network because of security issues such as firewall. There is a need for a framework to support HLA-based distributed simulations on Grid.

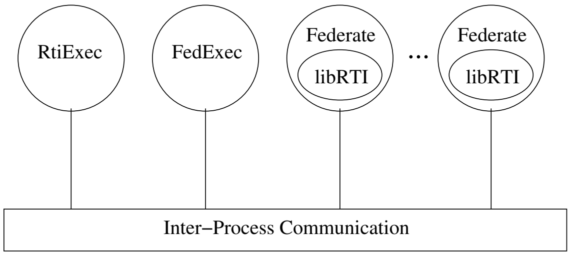

Figure 1: RTI components.

In this paper, we propose a distributed simulation backbone for executing HLA-based simulation over the Internet. The backbone makes use of a Federate-Proxy-RTI architecture, in which a remote proxy acts on behalf of the federate in interacting with RTI. The simulation federate runs at the client side, which ensures the security of the simulation model. This architecture hides the communication over the Grid network, and provides user transparency and simulator reusability. It also facilitates migration of federate code without affecting other parts of the simulation, and the simulation backbone itself can also be migrated, as it is not involved in the simulation logic. Moreover, the backbone, implemented in Java, allows simulators from heterogeneous platforms to join into a single simulation over the Internet. We have designed and implemented a Grid-enabled HLA API based on DMSO's RTI 1.3 implementation, and the Grid system runs the Globus Toolkit Version 3.

Zajac, et al. [11] proposed a system to enable HLA-based simulations on Grid, but they focused 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.

Morse, et al. [7] proposed an architecture using the Extensible Modeling and Simulation Framework (XMSF) compliant Web Services for web based federates communicating with RTI. Their approach is based on formatting the RTI calls via Simple Object Access Protocol (SOAP) [8] and employing the Blocks Extensible Exchange Protocol (BEEP) [1] communication layer to enable bi-directional calls and callbacks via web service. Our proposed backbone is different from Morse's approach, because the Proxy in our backbone is implemented using Grid service, and provides standard Grid service mechanisms for state management, notification, and so on, while traditional web service does not have.

Chen, et al. [2] proposed a decoupled federate architecture for distributed simulation cloning. In their architecture, the simulation model is separated from the runtime component by using the concept of virtual federates interlinking corresponding physical federates. The decoupled federate architecture shares some similarities with our design.

The rest of the paper is organized as follows: Section 2 outlines distributed simulation using HLA. Section 3 discusses the design of the Federate-Proxy-RTI architecture. Section 4 discusses our prototype implementation. In Section 5, we conclude with a brief summary and discuss our plans for future work.

2 HLA-based Distributed Simulation

In HLA-based distributed simulations, the Run-Time Infrastructure (RTI) provides common services during the execution of an HLA simulation by implementing the HLA interface specification. RTI consists of RTI Executive process (RtiExec), Federation Executive process (FedExec) and the libRTI library, as illustrated in Figure 1.

The RtiExec manages multiple federation execution with different names within a network. The FedExec manages multiple federates within the 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 federate code

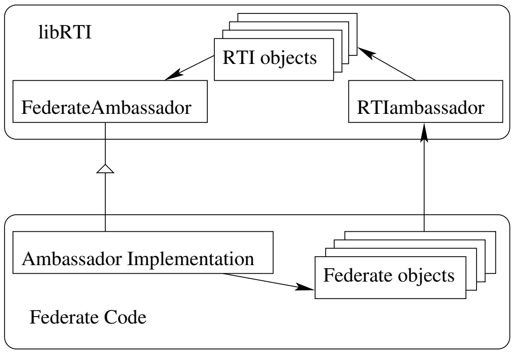

Figure 2: Federate code communicate with libRTI through method calls of *RTIambassador* and the callback functions provided in the *FederateAmbassador* class.

should implement the abstract class *FederateAmbassador* to provide callbacks functions. Figure 2 shows how federate codes communicate with libRTI.

HLA uses the object oriented model, in which each federate must be part of a class 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 have. 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 only relevant to that very instance in time and it is not stored.

3 Design

3.1 Overview

The backbone is designed to enable distributed simulation using HLA on Grid, and our goals are:

- to provide a standard HLA API as defined by the DMSO implementation, 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 backbone.

In our design, different participants (clients) in the same simulation run their federate codes at their local sites, and 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 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 providing the standard DMSO HLA API to the federate codes, is implemented to translate the communications into Grid services invocations. Additional Grid services are needed to support the creation of RTI, discovery of federations, etc.

3.2 Detailed Design

The framework consists of two major components: client , resource and supporting Grid services, and they are interconnect through the Grid network, as shown in Figure 3.

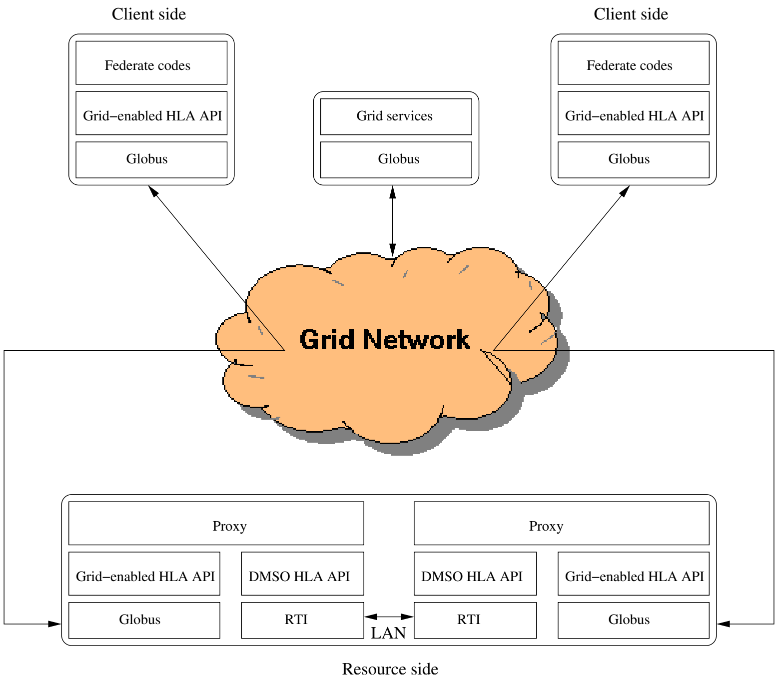

Figure 3: Architecture of Proxy-based HLA simulation on the Grid.

3.2.1 Client Side

The client side provides the standard DMSO HLA API to the federate code, while allowing communication through Grid. We make use the Globus Grid middleware as the lower level communication channel, and provide a Grid-enabled API to translate federate-RTI communication into Grid service innovations, as illustrated in Figure 3.

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 Grid are achieved using remote Grid service invocations by embedding the parameters inside invocations. When there is a *FederateAmbassador* callback from the RTI, a notification will be generated at the proxy site and delivered to the client of the callback together with the parameters needed for the callback. Figure 4 provides the conceptual view of the interactions between client side and the proxy.

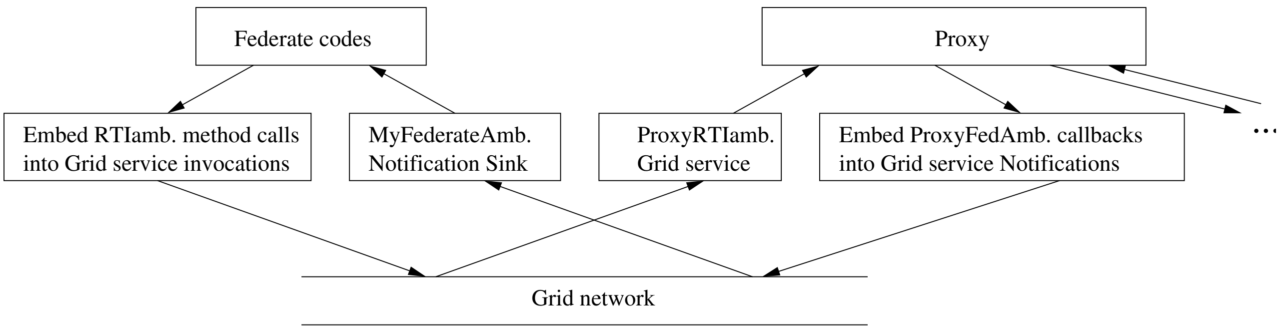

Figure 4: The interaction between client side and the proxy in the framework.

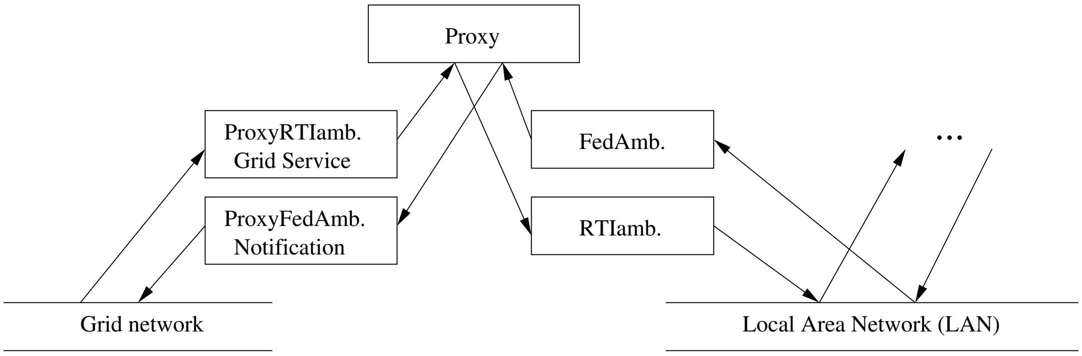

Figure 5: The interaction between the proxy and the RTI at the resource side.

3.2.2 Resource Side

At the resource side, the *proxy* acts on behalf of the client and communicates with the RTI through Local Area Network (LAN), as shown in Figure 5. 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 notifications.

3.2.3 Other Grid services

Besides the Grid services to enable the communication between client's federate code and the remote RTI, other Grid services are required for creating RTI, discovering 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.

4 Implementation

A prototype has been implemented based on the Globus Toolkit version 3 (GT3). We have implemented the *RTIambassador* services' API for Federation Management, Time Management, Object Management, Declaration Management, and Ownership Management. As mentioned in [7], to implement the Distributed Data Management (DDM), we need to keep state and context for each region, and for all the dimension handles associated with it within our translation libraries at runtime. We plan to implement the DDM in future.

4.1 Client Side

At the client side, three main classes are being implemented, as shown in Figure 6:

- *HLAGridNullFedAmb* class extends the default *NullFederateAmbassador* class provided in the DMSO's HLA implementation, and all the interfaces remain the same.
- *ProxyListerner* class acts as a notification sink. Whenever, a notification is delivered from the resource side to the client side, the *deliverNotification* method of the *ProxyListerner* is called to notify the callback of *HLAGridFedAmb*.
- *HLAGridRTIAmb* embeds all the *RTIambassador* calls into a Grid Service invocation.

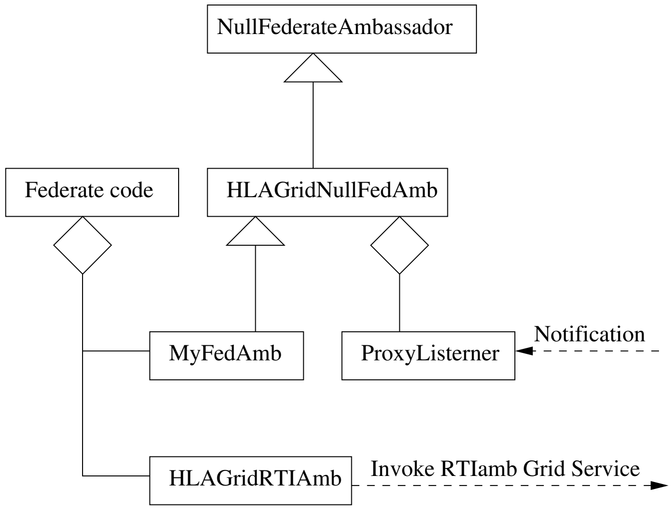

Figure 6: The class diagram of the client side.

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

4.2 Resource side

At the resource side, five main classes are used:

- *ProxyServiceProvider* class implements the *OptionProvider* and *GridServiceCallback*. It accepts remote invocations of *RTIambassador* method calls, and delivers *FederateAmbassador* callbacks to the client.
- *ProxyRTIComponent* contains a *RTIambassador* to communicate with the RTI.
- *ProxyFedComponent* also contains a *ProxyNullFedAmbassador*, which embeds callbacks into Grid Service notifications.

4.3 HLA Simulation on Grid Walk-through

We describe the detailed steps involved in a HLA simulation on 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 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.
  2. Create federate ambassador and RTI ambassador: the *MyFedAmb* and *HLAGridRTIAmb* will be instantiated at the client side. The *RTIAmbassador* and *ProxyNullFedAmbassador* will be instantiated at the resource side.
  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. Note that, steps 1 to 3 have to be done atomically, because otherwise federates belonging to the same federation could create several RTI instances and federations. Atomicity is handled by the Index/RTI service and RTI itself in our implementation.

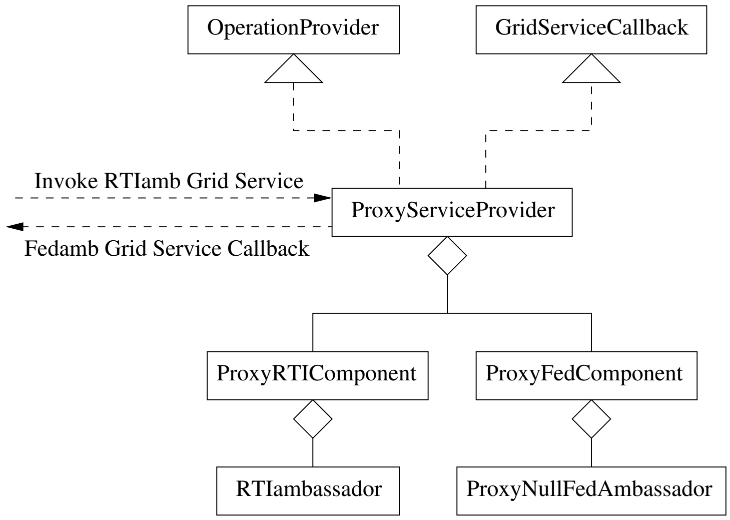

Figure 7: The class diagram of the resource side.

4. Initialize, publish and subscribe: all simulation settings will be initialized by invoking the *RTI-Ambassador* at the resource side.
5. Enable time constraints and time regulations: same as in step 4.

- Main loop: step 6

1. this is the main body of the federate code, which includes *RTIambassador* method calls and *FederateAmbassador* callbacks, as shown in Figure 8 and Figure 9.

- Shutdown: steps 7 to 9

7. Resign from federation: this involves a method call to remote *RTIAmbassador*.
8. Destroy federation execution: besides the Grid service invocation to destroy 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.

When federate executes *RTIambassador.tick()*, it passes the control to the RTI and wait for *TimeAdvanceGrant()* callback to advance the simulation time. Figure 10 depicts the procedures both the client interact with the proxy to execute the *tick()* method, and the detailed procedures are:

1. Client's federate code makes a method call *RTIambassador.tick()* to the remote proxy's RTI component.
2. Proxy's RTI component makes the actual *RTIambassador.tick()* call to the local RTI, and passes control to RTI.
3. RTI makes a *FederateAmabassador.discoverObjectInstance()* callback to the ProxyFedComponent, and the ProxyFedComponent generates a Grid service notification to client's *MyFedAmb*.
4. After *MyFedAmb* executes the *discoverObjectInstance()* method, it will notify ProxyFedComponent that *discoverObjectInstance()* method is done.
5. Repeat step 3 and 4 for *FederateAmabassador.reflectAttributeValues()* callback.
6. Repeat step 3 and 4 for *FederateAmabassador.timeAdvanceGrant()* callback.

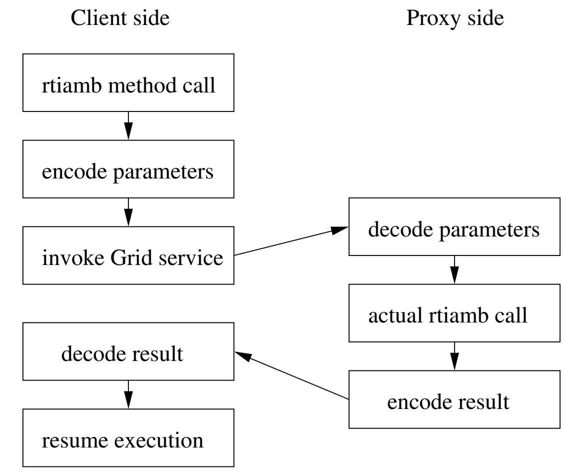

Figure 8: Executing a *RTIambassador* method call through Grid service invocation.

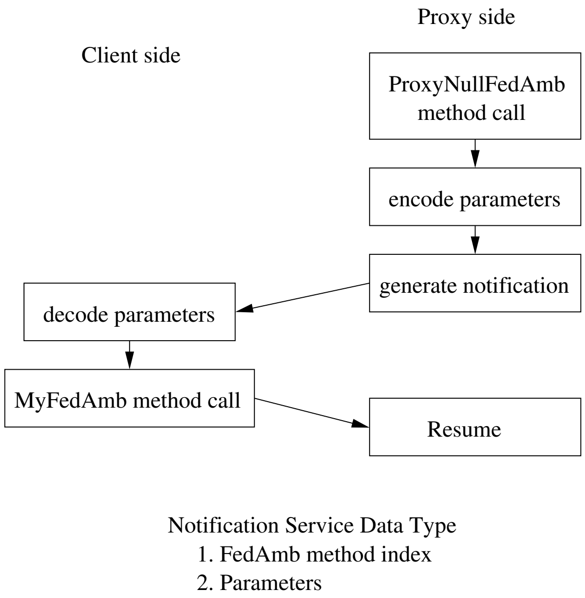

Figure 9: Executing a *FederateAmbassador* callback through Grid service notification.

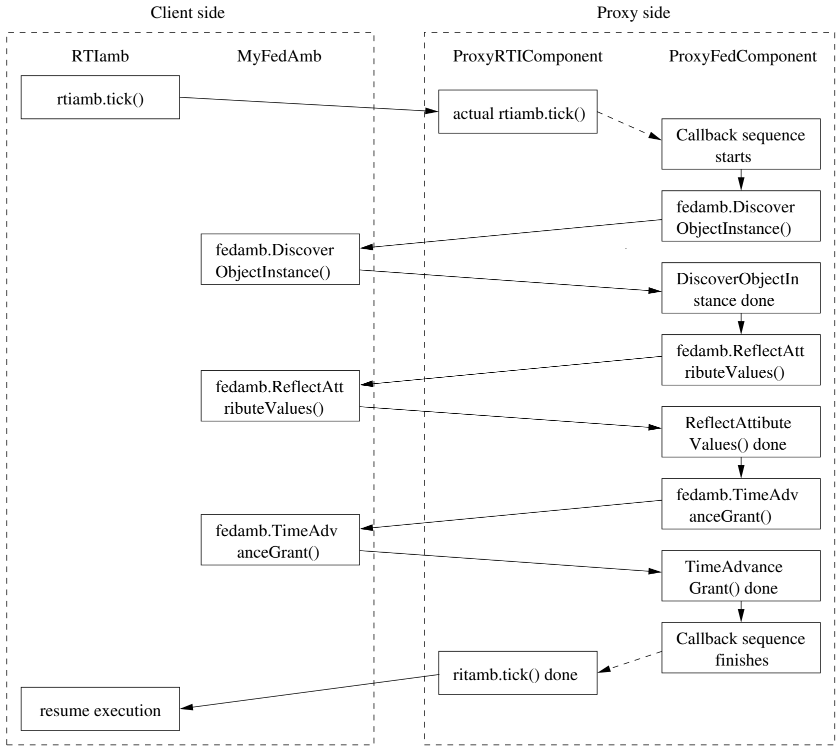

Figure 10: Interactions between the client and the proxy in a sample invocation of *RTIambassador.tick()*.

7. Control is passed back to the ProxyRTIcomponent to resume execution, which ends the *tick()* method call.

8. The Client's federatecontinuess its execution.

Note that, both the RTIAmbassador method call and FederateAmbassador callbacks are handled in a blocking manner, which ensures the correct order of method calls, and guarantees all the events interchange among time regulating and time constrained federates in a correct and causal manner [2].

Moreover, there are some detailedimplementationn issues need to be mentioned. In the DMSO's HLA/RTI implementation, the Java Binding only provides a JNI interface for making calls to objects without a proper constructor. For example, in the C/C++ version, instances of the *AttributeHandleValuePairSet* class can be constructed by the *AttributeSetFactory* class, but it is not supported in the Java version. As a result, we have implemented our own classes with constructors to provide all the API specified in the HLA/RTI documentation, such as the *ReflectedAttribute* class. Also, the Distributed Data Management (DDM) is not implemented, as itrequiress keeping the state and context for each *region* and for all the dimension handles associated with it at runtime, and similar issue was brought up in [7] as well.

A prototype is implemented in Java, and the Grid system runs the Globus Toolkit version 3. Preliminary testing is done by varying the granularity of the communication like changing the attribute size in *RTIambassador*'s *UpdateAttributeValues* method call. It shows that translating communication between federates and RTI into Grid service invocations and notifications incur some overhead, but with the increase of granularity of communication, it performs reasonably well.

5 Conclusion and Future Work

In this paper, we propose a framework for distributed simulation using the High Level Architecture (HLA) over the Grid. The framework makes use of a Federate-Proxy-RTI architecture, in which a remote proxy to act on behalf of the federate in interacting with the RTI. This architecture hides the communication over the Grid network, and provides user transparency and reusability. Prototype is implemented based on the DMSO's HLA implementation and Grid service platform provided by the Globus Toolkit version 3 to achieve compatibility and interoperability. More testing are needed such as tests for latency and synchronization performance.

References

[1] BEEP. Blocks extensible exchange protocol, 2003.

[2] D. Chen, S. J. Turner, B. P. Gan, W. Cai, and J. Wei. A decoupled federate architecture for distributed simulation cloning. In *Proceedings of the 15th European Simulation Symposium and Exhibition*, 2003.

[3] J. S. Dahmann, F. Kuhl, and R. Weatherly. Standards for simulation: as simple as possible but not simpler the high level architecture for simulation. In *Simulation*, pages 378–387, June 1998.

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

[5] 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.

[6] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[7] 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*, 2003.

[8] SOAP. Simple object access protocol, 2003.

[9] S. Tuecke, K. Czajkowski, I. Foster, J. Rey, F. Steve, and G. Carl. Grid service specification, 2002.

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

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