Introducing Web Services in HLA-Based Simulation Application

***Abstract* – High Level Architecture (HLA) conduces to the integration of different models to form a complicated distributed simulation system. However, there exist some limitations in HLA-based simulation application in some aspects. In the paper, aiming at resolving these problems, we focus on the study of introducing web services in HLA-based simulation application and propose Web Services-Based HLA Simulation Framework (WSHLA). Firstly, the structure of and the web services in WSHLA are presented. Then, the implementation of every components and the execution flow of the framework are discussed respectively. Experimental results validate the framework and demonstrate that using the framework to introduce web services in HLA-based simulation application can effectively make up for its limitations and ensure its better running at expense of some time.**

I. INTRODUCTION

High Level Architecture (HLA), which is designed to promote standardization in the Modeling and Simulation (M&S) community and to facilitate the reusability and interoperability of M&S components, is an advanced, distributed high-level simulation architecture, which may be used to integrate models in different domains together to form a complicated simulation system [1].

However, there exist the following limitations in HLA-based simulation application [2].

In HLA, federates and RTI communicate with each other directly using the address and the port specified in RTI Initialization Data (RID) file. When an HLA-based simulation application is running in the environment protected by firewall, its communication will be blocked by firewall, which even makes the whole simulation fail.

Due to the characteristics of programming language-specific and platform-specific of Run-Time Infrastructure (RTI), which is the software that implements the HLA Interface Specification (IFSpec) and provides common services to simulation system, the federate programmed by one language (or using one RTI) cannot work well with the federate programmed by another language (or using another RTI). So, the transplant of federates is difficult and the reusability of HLA-based federates is not good enough.

HLA, initially designed for military purpose, doesn't take the standards or technologies of other domains into account adequately. So, HLA may not be well compatible with those technologies. That may be regarded as the main reason of its poor interoperability in multi-domains.

The above limitations make it difficult for HLA to get an in-depth development. Therefore, HLA simulation system should absorb and assimilate other related standards or technologies to achieve better development.

Using eXtensible Markup Language (XML) and HyperText Transportation Protocol (HTTP), web services represent a new distributed computing pattern. Web services, which allow applications to communicate in a platform-independent and programming language-independent manner, is used to build loose-coupling distributed applications with good compatibility. More importantly, due to the use of HTTP, these applications may not be blocked by firewall at all.

According to the above analysis, if web services can be introduced in HLA-based simulation application, the limitations mentioned above will be resolved effectively, which will greatly improve its reusability and interoperability. However, because web services and HLA are different standards for different purposes and domains, it's difficult to glue the two together. Therefore, it has been a hot research to find a proper and effective approach.

According to the characteristics of HLA-based simulation application and web services, we propose Web Services-Based HLA Simulation Framework (WSHLA) and use the framework to introduce web services in HLA-based simulation application. Firstly, the structure, including all components such as RTI side and client side, and web services are presented in detail. Then, the implementation of every components and the execution flow of the framework are discussed respectively. Experimental results validate the framework and demonstrate that using the framework to introduce web services in HLA-based simulation application can effectively make up for its limitations and ensure its better running at expense of some time.

The rest of paper is organized as follows: Section 2 presents literature review. Section 3, 4, 5 discusses the structure of, web services in and implementation of WSHLA one by one. Section 6 validates the framework through experiments and analyzes experimental results. Section 6 concludes the paper and gives the plan of future work.

II. LITERATURE REVIEW

In current literature, there are three methods of introducing web services in HLA-Based simulation application: extending HLA federates, extending HLA communication tier and web-enabled RTI (WE RTI) [3].

(1) Extending HLA federates. The method, which just applies web services to all federates in a federation without

changing RTI, is the simplest method in the above three methods. David Macannuco uses the method in Ref. [4] to develop a distributed simulation and training system that provides a common operating environment or user interface to bind selected components together and allows any user to access and manage it using a web browser. Though with a simply structure, the method needs different XML schemes specified by users when running in different environment. So, the extendibility of the method is not good enough.

(2) Extending HLA communication tier. The method changes all HLA services in IFSpec to web services. RTI and federates may use these web services instead of original HLA services to communicate. Though it sounds well, the method is actually very difficult to be carried on because 1) the whole RTI needs to be redesign and 2) when the new RTI is put to use, users must modify original federates to adapt to the change. Bjorn Moller describes in Ref. [5] the design of HLA Evolved WSDL API, which is still far from practical use.

(3) WE RTI. Because of the difficulty of the second method, some scholars proposed web-enabled RTI that can half extend RTI communication tier using web services. The goal is to enable a simulation to communicate with an HLA/RTI through web-based services. The long-term goal of it's to have multiple federates that can reside as web services at Wide Area Network (WAN), permitting an end-user to compose a federation from a browser. In Ref. [6][7], Katherine L. Morse builds a prototype HLA federation using WE RTI. In that federation, Simple Object Access Protocol (SOAP) and Blocks Extensible Exchange Protocol (BEEP) are used for communication between two federates. Compared with the first method, WE RTI has a better universality and can be applied to existing simulation applications easily. However, the introduction of bridge mechanism, which is still immature, makes it a little difficult and complicated to put the method into practice.

In a word, the first and the third are two comparatively practical methods. A framework, which combines both the advantages of the two methods, was proposed by us in Ref. [8]. The framework, with a simple structure and good universality, can effectively introduce web services in HLA-based simulation application. But, the framework still has its limitations, such as low efficiency. In the paper, we will focus on improving further the initial framework.

III. STRUCTURE

From the above analysis, we may safely arrive at the conclusion that the old manner of communication is the main reason resulting in those limitations. So, we propose Web Services-Based HLA Simulation Framework (WSHLA) and use the framework to introduce web services in HLA-based simulation application to change the situation.

A. Overview

WSHLA may be divided into two main components (Client Side and RTI Side) structurally. The overall structure of WSHLA is shown in Fig. 1.

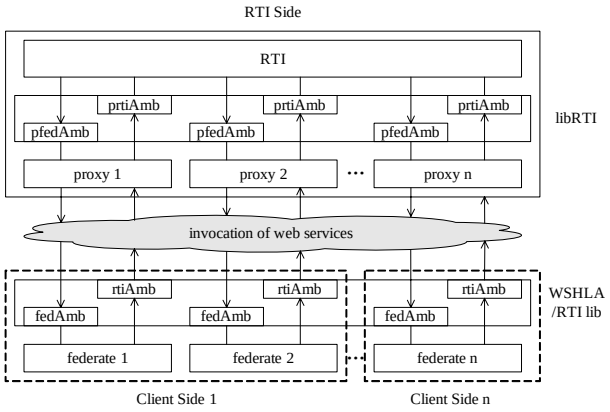

Figure 1. Overall Structure of WSHLA

In Fig. 1, there are one RTI side and some client sides. We deploy RTI at RTI side, and federates at client side. In RTI side, pfedAmb is the federate ambassador of HLA/RTI, and prtiAmb is the RTI ambassador of HLA/RTI. In client side, conforming to IFSpec, WSHLA/RTI lib is designed to call web services. FedAmb is the federate ambassador of WSHLA/RTI, and rtiAmb is the RTI ambassador of WSHLA/RTI. Proxy is a new introduced object, which will be discussed at length in the next paragraph.

B. RTI Side

In WSHLA, RTI and all necessary web services are deployed at RT Side. Besides the two parts, proxy, a new introduced object, is also deployed at RTI Side.

Definition 1: In WSHLA, the object, deployed at RTI side, which is on behalf of a federate in communicating with RTI, is called proxy.

In WSHLA, every federate in client side has a corresponding proxy in RTI side. Federates and RTI communicate indirectly via the proxy using web services other than communicate directly, which means the proxy acts on behalf of a federate and communicates with RTI.

Before a federate joins a federation execution, it calls a web service to create a corresponding proxy in RTI side, and then the proxy not the federate joins the federation execution. From the point of HLA, the proxy may be regarded as a special federate. When calling a RTI ambassador service, a federate calls a web service to send the request to its corresponding proxy, which will then transfer the request to RTI. Similarly, when calling a federate ambassador service, RTI also sends the request to a proxy, which will then transfer the request to its corresponding federate using web services.

C. Client Side

In WSHLA, simulation applications, federates and models run at client side. Due to the introduction of web services, federates running at client side don't call the *libRTI* provide by RTI software, but call the custom WSHLA/RTI library provided by WSHLA. Except for a few special services of WSHLA, the interface of two libraries is almost identical,

which makes the conversion from HLA-base simulation application to WSHLA-based application easily and smoothly.

D. Advantages

In WSHLA, RTI and federates communicate with each other indirectly via the proxy using web services. The invocations of web services, which use XML and HTTP, won't be blocked by firewall. In addition, proxy is deployed at the same computer where RTI resided at, therefore the communication between RTI and proxies won't be blocked by firewall either. So, WSHLA-based simulation won't be blocked by firewall at all.

Because federates call WSHLA/RTI lib other than *libRTI*, and WSHLA/RTI lib is just the encapsulation of the invocations of web services, any program languages that can call web services can be used to program simulation federates in different platforms. So, the reusability of federate will be improved greatly.

Meanwhile, because web services has been well accepted in a lot of domains, we may use web services as a bridge to connect HLA and other related standards and technologies, and then to ameliorate the interoperability of HLA-based simulation application.

IV. WEB SERVICES

The greatest improvement of WSHLA is the introduction of web services that are responsible for the communication between federates and RTI. Improper design of web services will result in low efficiency of WSHLA, even the failure of the whole framework. Therefore, the design of web services is essentially important.

HLA services in IFSpec can be divided into two categories: RTI ambassador services and federate ambassador services (or callbacks), which are used by RTI and federates to communicate. However, in WSHLA, RTI and federates communicates indirectly via the proxy using web services. Therefore, web services should take the responsibility for communication. So, firstly, we must design web services for the two categories of HLA services. Meanwhile, due to the introduction of web services and the change of structure, it's necessary to introduce some auxiliary services to adapt to these changes.

Therefore, the web services in WSHLA should comprise three categories: RTI ambassador web services, federate ambassador web services and auxiliary web services.

(1) RTI ambassador web services

RTI ambassador services, provided by *libRTI*, are stable and don't change with the change of federates. The characteristic simplifies the design of this category of web services. What we should do is to use web services to encapsulate each RTI ambassador service and deploy these web services at RTI side. Therefore, each RTI ambassador service has its corresponding web services. When calling a RTI ambassador service, users may call the corresponding web services directly. Of course, the invocations of these web services can be encapsulated again for facilitating their use.

(2) Federate ambassador web services

Compared with the previous work, it's a little difficult to design federate ambassador web services. Because, it's user not *libRTI* who provides federate ambassador services for each federate, which may run at different client sides. If we apply the previous method to this category of web services, federate ambassador web services may be deployed at different client sides, which will then make the task of designing proxy and using WSHLA inconvenient (Because, in the circumstance, each client side must install web services container, which is sometimes beyond the capability of users.).

We adopt timer and sharable file in our initial work to design federate ambassador web services. Though the method can run correctly, because timer is a resource-consuming object and reading/writing file are also time-consuming operations, the efficiency of the whole simulation system is low.

In this paper, we adopt message mechanism. When a proxy receives a request for a callback, it will encapsulate the information about the callback, such as name, parameters and etc, as a message that will then be send to a message queue. Meanwhile, a web service named *GetMessage* is provided for users to get those messages that interest them. After receiving messages, users may actually execute their custom federate ambassador services.

(3) Auxiliary web services

Because of the change of structure, WSHLA also need to support some auxiliary operations, such as: starting RTI, shutting down RTI and etc. Therefore, we design the following web services for these operations.

Starting RTI: Before running a simulation, users call the web service to start RTI.

Getting RTI running status: Before starting RTI, users call the web service to judge whether RTI is running in RTI side or not.

Shutting down RTI: After all simulations are end, users call the web service to shut RTI down.

Getting federation execution status: Before shutting down RTI, users call the web service to judge whether there are federation executions in RTI side or not.

V. IMPLEMENTATION

A. RTI Side

The main functions of RTI side include: to provide all necessary web services, to manage proxy such as creating a proxy, destroying a proxy, RTI communicating with proxy and etc. So, we design six important classes, which are shown in Fig. 2.

FedAmbImpl, implementing *NullFederateAmbassador*, is a specific implementation of federate ambassador of proxy. Its main function is to receive callbacks called by RTI and to use *MessageSink* to manage the messages that encapsulate the information about these callbacks.

MessageSink encapsulates the information about the callback as a message and then sends the message to a message queue when a callback in *FedAmbImpl* is called by RTI.

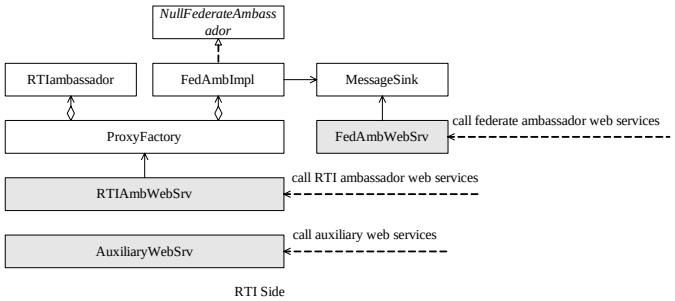

Figure 2. Class Diagram of RTI Side

FedAmbWebSrv provides a web service named *GetMessage* that can be used to get message from a message queue.

ProxyFactory, aggregating instances of *FedAmbImpl* and *RTIambassador*, manages the proxy, such as creating a proxy, destroying a proxy and etc.

RTIAmbWebSrv, owning an instance of *ProxyFactory*, provides RTI ambassador web services to users.

AuxiliaryWebSrv provides auxiliary web services, such as starting RTI, shutting down RTI and etc to users.

B. Client Side

The main function of client side is to encapsulate the invocation of web services and provide standard DMSO (Defense Modeling Simulation Office) HLA interface, including RTI ambassador interface and federate ambassador interface. So, we design three important classes, which are shown in Fig. 3.

WSHLA_RTIAmb: Though users can call web services in their simulation program directly, it may be inconvenient for those who know little about web services. Therefore, we encapsulate the invocations of RTI ambassador web services and auxiliary web services and provide standard DMSO HLA interfaces to users. Hiding the direct invocations of web services, the class makes programming much easier. The standard APIs also enable users to be more consistent with their programming habit and simplify the conversion from HLA-based simulation application to WSHLA one. Due to the characteristics of platform-independent and programming language-independent of web services, different programming languages can be used to encapsulate these web services in different platforms to extend its application area.

WSHLA_NullFedAmb, implementing *NullFederateAmbassador*, uses *GetMessage* service to get the information about a callback, and then executes the callback actually.

MyFedAmb inherits *WSHLA_NullFedAmb*. Users may override the callbacks according to their needs.

C. Execution Flow

Introducing web services in HLA-based simulation application makes its execution flow different from the original application. We will discuss the execution flow from the two aspects of calling RTI ambassador web services and calling federate ambassador web services.

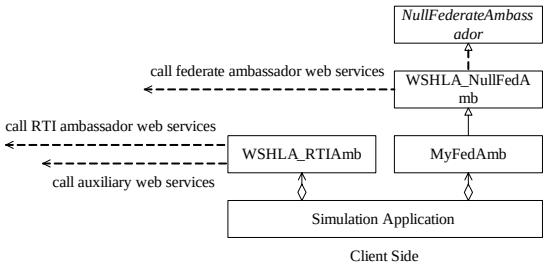

Figure 3. Class Diagram of Client Side

(1) RTI ambassador web services

The application in client side sends a request for calling a RTI ambassador service using web service. After receiving the request, RTI side parses the request. If the request is *joinFederationExecution*, a corresponding proxy is created in RTI side and then joins federation execution. If not, the corresponding proxy sends the request to RTI for executing. When the service is finished, the proxy returns the result to client side. The execution flow of calling RTI ambassador services is shown in Fig. 4.

(2) federate ambassador web services.

After a callback is created by RTI, a proxy will encapsulate the information about the callback and then send to client side as the form of message. Client side receives the message and executes it. The execution flow of invoking federate ambassador services is shown in Fig. 5.

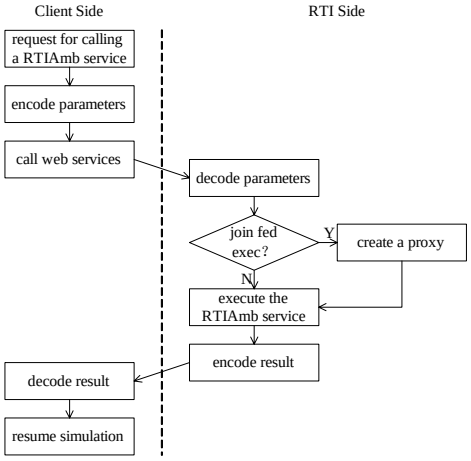

Figure 4. Execution Flow of Calling RTI Ambassador Services

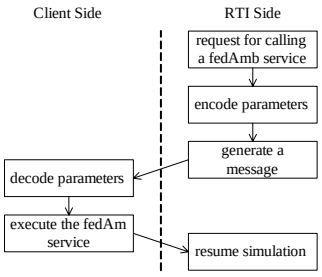

Figure 5. Execution Flow of Calling Federate Ambassador Services

VI. EXPERIMENTS

Experiments are designed to validate WSHLA. Experimental environment is: Windows XP Professional, Java 1.4, C#, Pitch pRTI 1.3, Tomcat 4.1 and Axis 1.4. The experimental system is shown in Fig. 6.

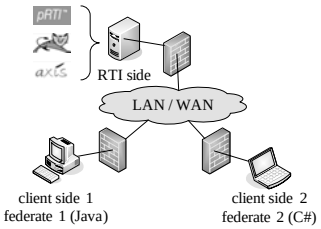

Figure 6. Structure of Experimental System

In the experiment, a federation comprising two federates programmed with Java and C# respectively is created. Running at client side 1 and client side 2 respectively, the two federates have the capability of sending/receiving interactions and updating/reflecting attributes.

We modify the benchmark program from DMSO HLA packet and focus on the latency benchmark [9][10]. 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/receive an interaction or update/reflect an attribute.

We measure the elapsed time in LAN and WAN with firewall turned on and off. Meanwhile, we also conduct the same experiment in the same circumstance using HLA and our initial prototype system for comparing the experimental results. The experimental result measured in LAN and in WAN is shown in Fig. 7 and Fig. 8 respectively.

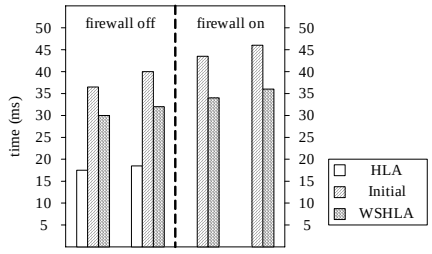

Figure 7. Experimental Result in LAN

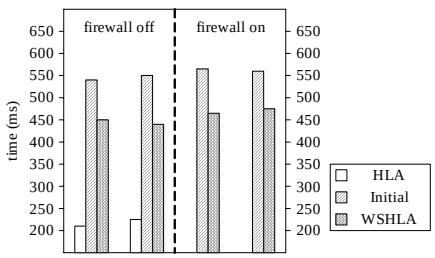

Figure 8. Experimental Result in WAN

First, from the experimental result, it's obvious that when firewall is turned on, HLA-base simulation application is completely blocked by firewall and we cannot get any data in this circumstance. On the contrary, firewall exerts no influence on any version of WSHLA-base simulation application.

Second, the efficiency of current version of WSHLA is better than the initial version. Because, timer is a resource-consuming object and reading/writing file are also time-consuming operations. The two reasons make the efficiency of the whole system low. In the current version, message mechanism allows applications just listen to messages without consuming extra resource and operations. So, the efficiency of the new version of WSHLA is improved.

Third, federates programmed with Java and C# can work together successfully; so did those programmed by other languages that can invoke web services. The characteristic will increase the reusability of federates and flexibility of WSHLA greatly.

However, in the same circumstance, ant version of WSHLA-based simulation application will cost much more time than HLA-based simulation application. Because, the size of soap packet used in WSHLA is much larger than the packet used in HLA. In addition, when a service is requested or responded, the necessary parameters will be marshaled or unmarshaled, which will decrease the efficiency of WSHLA. The above reasons increase the traffic of simulation application and make its efficiency lower. However, it's an inevitable limitation of introducing web services in HLA-based simulation application.

VII. CONCLUSION

For the purpose of making up for the limitations existing in HLA-based simulation application, we propose a framework named WSHLA and use the framework to introduce web services in HLA-based simulation application. With the help of web services, we can build a loose-coupling simulation system, which will increase the reusability, interoperability and flexibility of HLA-based simulation application. Experimental results, which validate the framework, demonstrate that using the framework to introduce web services in HLA-based simulation application can effectively make up for its limitations and ensure its better running at expense of some time.

The biggest problem existing in WSHLA, no matter what version is given, is the efficiency of the framework is lower than HLA. So, we will continue to find a better way to improve the efficiency of WSHLA, and then promote further the use of WSHLA.

REFERENCES

[1] Defense Modeling Simulation Office, "High Level Architecture Run-Time Infrastructure RTI 1.3-Next Generation Programmer's Guide Version 5," http://www.dmso.mil/, 2002.

[2] D. Brutzman, M. Zyda, J. M. Pullen, K. L. Morse, "Extensible Modeling and Simulation Framework (XMSF) Challenges for Web-Based Modeling and Simulation," XMSF 2002 Findings and Recommendations Report: Technical Challenges Workshop and Strategic Opportunities Symposium, Monterey, 2002.

[3] C. Han, R. Ju, K. Huang, "HLA Simulation System Extension Based on Web Services," Computer Engineering, vol. 32, pp. 20-22, December 2006.
[4] D. Macannuco, K. B. Donovan, M. Falash, L. Salemann. "A Web-based Infrastructure for Simulation and Training," Proceedings of Fall Simulation Interoperability Workshop, Orlando, 2004.
[5] B. Moller, C. Dahlin, "A First Look at the HLA Evolved Web Service API," Proceeding of the 2006 European Simulation Interoperability Workshop, Stockholm, 2006.
[6] K. L. Morse, D. L. Drake, R. P. Z. Brunton, "Web Enabling HLA Compliant Simulations to Support Network Centric Applications," Proceedings of the 2004 Symposium on Command and Control Research and Technology, San Diego, 2004.
[7] K. L. Morse, D. L. Drake, R. P. Z. Brunton, "Web Enabling an RTI – an XMSF Profile," Proceedings of the 2003 Europe Simulation Interoperability Workshop, Stockholm, 2003.
[8] H. Zhu, G. Li, L. Yuan, "WSHLA: Web Services-Based HLA Collaborative Simulation Framework," Proceedings of the 4th Collaborative Design, Visualization and Engineering, Shanghai, 2007.
[9] P. Knight, R. Liedel, "Analysis of Independent Throughput and Latency Benchmarks for Multiple RTI Implementations," Proceedings of 2002 Fall Simulation Interoperability Workshop, Orlando, 2002.
[10] M. Lorenzo, "RTI Latency Testing over the Defense Research and Engineering Network," Proceedings of Spring 2001 Simulation Interoperability Workshop, Orlando, 2001.