Advanced Engineering Forum *Online: 2012-09-26*
ISSN: 2234-991X, Vols. 6-7, pp 887-894
doi:10.4028/www.scientific.net/AEF.6-7.887
© 2012 The Author(s). Published by Trans Tech Publications Ltd, Switzerland.

PaaS RTI-Supporting Distributed Simulation Interaction in Cloud Computing Age

Huiyong Zhang [1, a], Caibin Liu [2,b] and Xiaowen Bi [3,c]

[1,2,3]Departement of Radar, Wuhan Ordnance N.C.O Academy of PLA, Wuhan 430075, China

[a]guanjuan1982@sina.com, [b]tilone@163.com, [c]blvccna@163.com

Keywords: HLA/RTI, Web Services, PaaS, Distributed Interaction Simulation.

Abstract. In Cloud Computing age, distributed interaction simulation would encounter many difficulties, especially the WAN-based interaction. A scheme of remodeling the traditional RTI software to be with "Platform as a Service (PaaS)" architecture was proposed. This paper puts forward two different software architectures to adapt to different network environments. Kernel idea is encapsulating the APIs of traditional RTI to be Web Services, deploying and scheduling these Web Services on servers, so, a distributed interaction platform is provided to simulation users on WAN. Users could utilize the services from the platform to execute simulation interaction. Finally, an experiment was carried out to test the time-latency performance of the remodeled software.

Introduction

In the M&S (Modeling and Simulation) field, the RTI software which implements the High Level Architecture (HLA) interface standard plays an important role in the distribution interaction simulation. Entering the Cloud Computing age, the study of distributed interaction simulation is expanding from LAN to WAN and traditional HLA/RTI faces many problems. For example: heterogeneous M&S environment needs cross-platform supporting software, WAN-based communication asks for reliable communication protocol and so on.

As a type of Cloud computing, Platform as a Service (PaaS) takes revolutionary changes to the software application. It facilitates deployment of applications without the cost and complexity of buying and managing the underlying hardware and software layers. The essence of the PaaS is transferring the computing resources from desktop to Internet, and managing these resources efficiently. The users of PaaS interoperate with the others using Web Services.

HLA and Web Services (WS) are two powerful architectures for information interoperability. The original purposes of them are not the same, and their working fields differ from each other. HLA is concerned with the simulation communication on LAN, while the WS is for WAN. In this paper, we would combine these two architectures, remodel the traditional RTI software to be with PaaS architecture (PaaS RTI), and deploy it on a data center. The users can invoke the PaaS RTI functions to implement then HLA simulation.

In literature 1, we have already proposed a simple PaaS RTI architecture, but it can only work in special WAN. In this paper, we will present a much more perfect one.

This paper is organized as follows. In section 2, we present the overview of the PaaS RTI. Section 3 and 4 describes two different structures to adapt to different network conditions. Section 4 introduces how to use the PaaS RTI. The performance experiment is executed in Section 5.

Paas Rti Overview

This paper combines HLA and Web/Grid Services, packages the traditional RTI API to be Web Services (RTI Web Services, RTIWS), deploys and schedules these Services on central servers, so a platform supporting distributed simulation interaction on WAN is set up. As figure 1 show, users invoke the RTIWS of the platform on central servers and a proxy federate member would be instantiated. This proxy member interacts with other proxy ones using HLA/RTI, and users communicate with its proxy federate member by Web Services. All the simulation data forwards through the central servers of the platform.

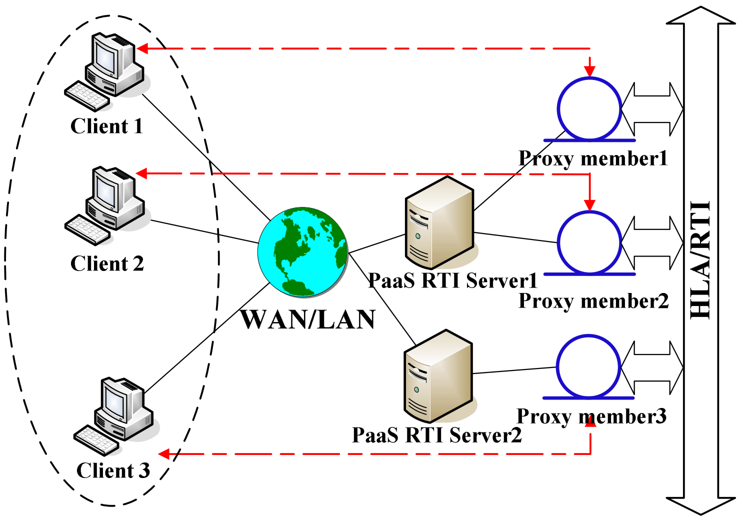

Fig. 1 Interaction through the PaaS RTI

Key technologies of the PaaS RTI include building RTIWS, scheduling and migrating RTIWS instances, and so on. The software architecture of PaaS RTI is described in figure 2.

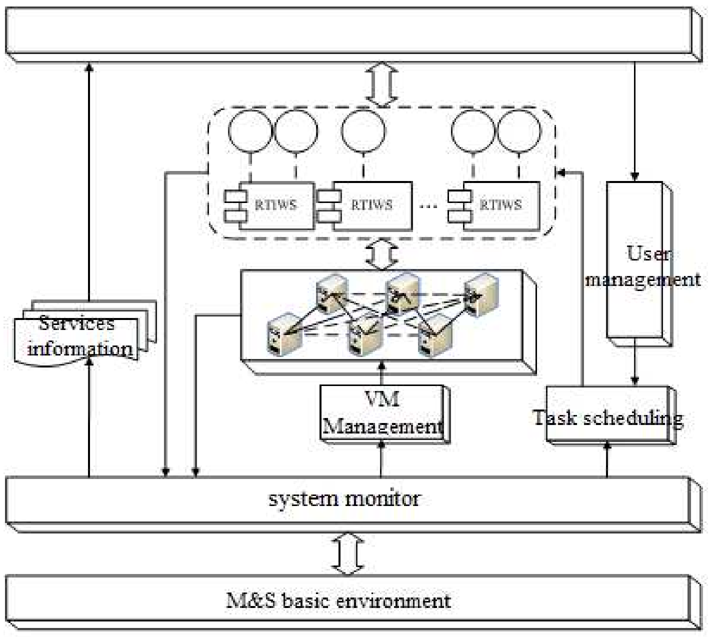

Fig. 2 Software Architecture of PaaS RTI

The seven main component modules of the platform listed below:

1. RTIWS: Web Services with RTI functions.
2. System monitor: querying and recording each physical server and virtual server data.
3. VM management: controlling the states of each virtual machine, such as power on, power office and so on.
4. Services information: recording the basic information of each RTIWS instance.
5. User management: managing the platform users.
6. Task scheduling: Dynamically and statically scheduling each RTIWS instance.
7. M&S basic environment building and management.

Basic RTIWS Constructing

RTIWS is the core of the system. Professor Wentong Cai and his research team have done many work in combining HLA and Web Services, and three types was put forward: Grid-facilitated, Grid-enabled, and Grid-oriented.

Both Grid-enabled and Grid-oriented can be chosen to constructing RTIWS. Grid-enable (also Web-enable) encapsulates the HLA/RTI API to be Grid/Web Services while Grid-oriented directly utilizes Grid/Web Services to deploy supporting platform following HLA standers.

RTIWS in PaaS RTI proposed in this paper would be a Grid-enable (also Web-enable) application. It is similar to WSDL API proposed by HLA-evolved.

The typical Web Services are usually "stateless", so when the RTI running, the state persistency and federates migration would be difficult using typical WS and much more additional work has to be done. Web Service Resource Framework (WSRF) specifies "stateful" Web Services [4], and it fits pretty nicely inside the whole Web Services Architecture.

The RTI software (pRTI1516 for example) includes two parts. The first part is the member methods of the RTIAmbassador class, and they can be invoked by the federates to request RTI services. The other is callback functions, which is invoked by the RTI itself to send message to the federate, and they are provided by the FederateAmbassador class.

WSRF (including WS-Notification) provides remote invoking and notifying. The client could invoke the services' WSRF interfaces, and the server can also notify the clients. The bidirectional communication mode meet communication needs of HLA. WSRF is selected to Package RTI to be WS.

Literature 1 proposed a basic RTIWS architecture (figure 3) which directly utilized the invoking and callback modes of WSRF to transmit message.

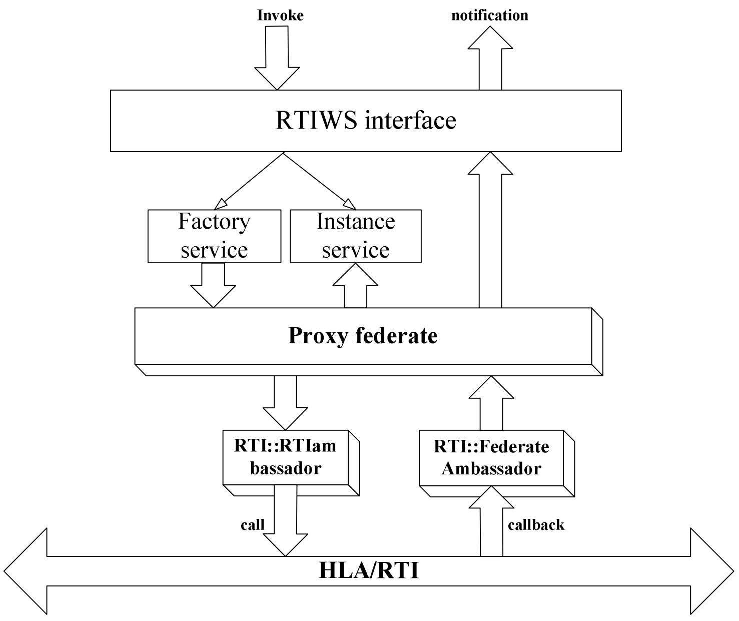

Fig. 3 Basic RTIWS Architecture

The Globus Toolkit 4(GT4) is provided by the Globus Alliance and it includes a complete implementation of the WSRF specification. We choose GT 4 as the WSRF tools to construct RTIWS. In the system, well known factory/instance patterns are adopted. The factory service creates resource, and the instance service provides the operations on the resource. The member functions of the RTIAmbassador class are translated into WSRF interfaces by embedding the parameters inside the invocations. Specially, if the parameter is of RTI-defined data type and the GT4 could not transmit it directly, it is necessary to convert the parameter to be normal data type. As for the callbacks of the federateAmbassador class from the RTI, GT4 will deliver the corresponding notification from server to the clients. The parameter of the notification usually is of String type. That is to say that we have to do some encoding and decoding work. Class architecture of BASIC RTIWS as shown in figure 4 is similar to the given description in literature 1.

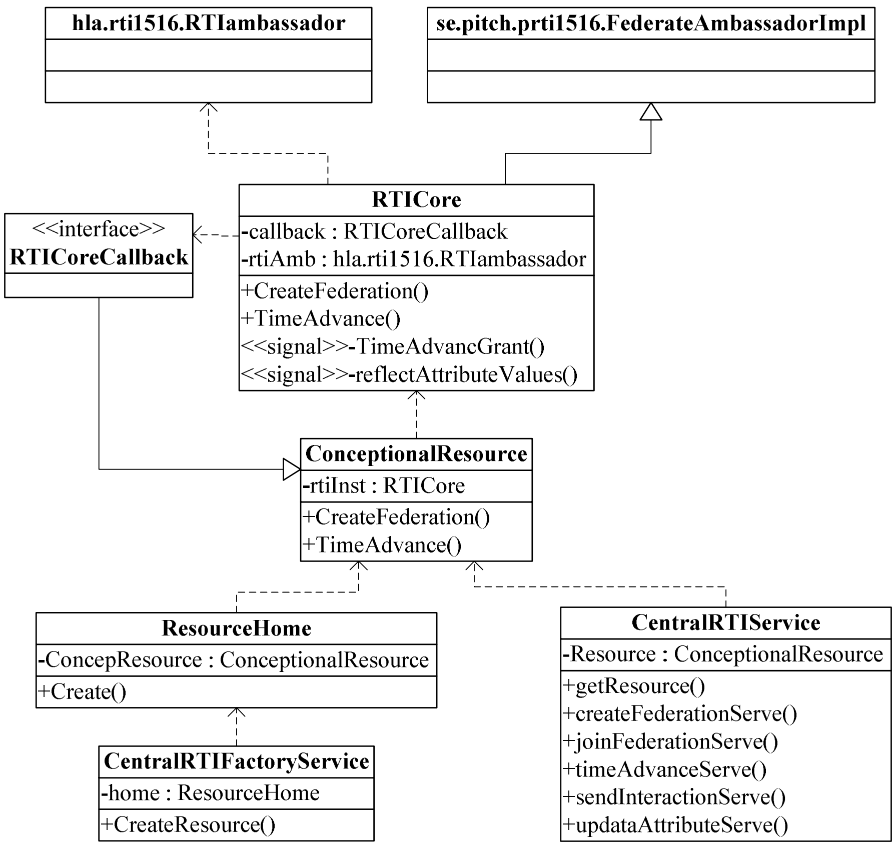

Fig. 4 Class diagram of Basic RTIWS

Extended Rtiws Constructing

Firewall policy is much stricter in many units and basic RTIWS as figure 3 and figure 4 shown cannot smoothly transmit message through firewalls on internet. Although client can invoke RTIWS through, RTIWS cannot send message directly to client, because GT4 implements WS-Notification utilizing RPC which not pass 80 port. So the basic RTIWS is adapted to special WAN, and towards WAN adopting strict firewall strategy, it cannot support the distributed simulation interaction. So we extend the basic RTIWS to solve this problem.

As figure 5 shows, using the extended RTIWS, a transmitting member is placed in the domain the client belongs to. In a domain, the firewall strategy is controllable, so WS-Notification message can be send to the client from the transmitting member. Another HTTP link is set up between RTIWS servers and the transmitting member. Callback information from RTIWS can reach a transmitting member in special domain, and then notify the client through WS-Notification. It is to say that there are two HTTP links between PaaS RTI servers and the domain of the client. One of the HTTP link is to transmit invoking message from client to server, while the other HTTP link is to send RTI callback message to client domain. Both of the HTTP link is of Web Services type, so it is not affected by the firewall of each domain.

The architecture of extended RTI is described in figure 6. Two Web Services, Transmitting Service and Notifying Service are deployed in the transmitting member. Transmitting Service is invoked by the RTIWS server to receive callback message, the message transports to Notifying Service through process communication, and Notifying Service Subscribe by the client would send the callback message to client immediately.

Difference from the basic RTIWS is the output way of the callback. Basic RTIWS directly send the callback message to client using WS-Notification, and extended RTIWS send the callback message to a transmitting member in the domain of the client through HTTP, and then, client would receive the callback message by WS-Notification.

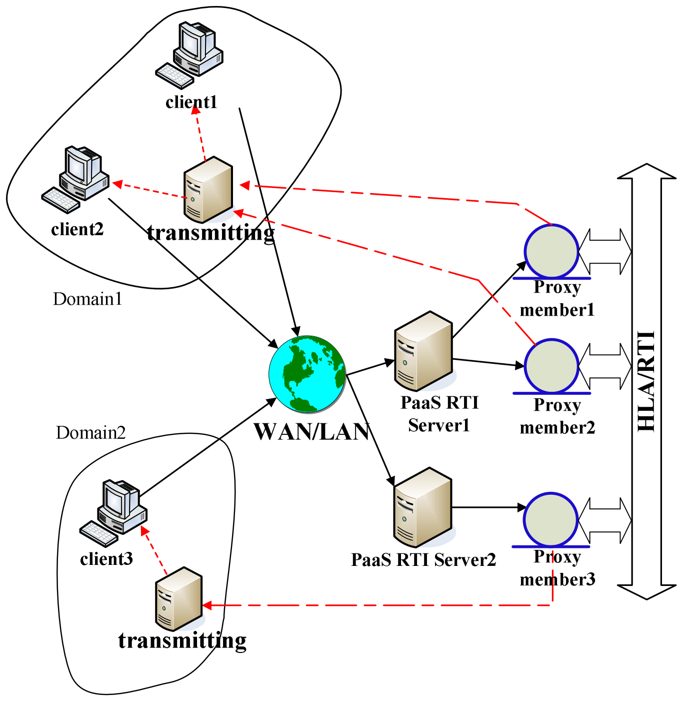

Fig. 5 Extended Interaction

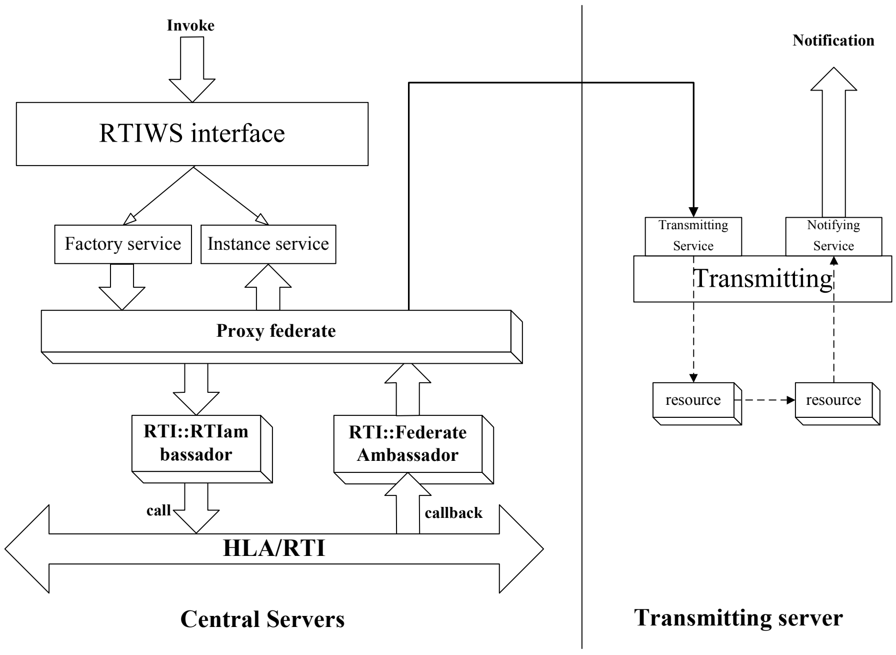

Fig. 6 Extended RTIWS Architecture

Performance Experiments

The PaaS RTI is based on the RTI software. Its performance especially the time-delay performance surely decreases. Our experiment is aimed to test the time-latency. We suppose that the network bandwidth is stable, the connection is reliable and the communication is unblocked. So the payload comes from the network can be neglected.

In the stage of interacting datas, the time-latency performance is mainly determined by the RTIWS component of the PaaS RTI. We take RTIWS perfonnance to present PaaS RTI.

The time-latency benchmark measures communication perfonnance. It usually refers to the elapsed time for federates to send and receive an attribute update.

The performance experiments execute under a simple WAN. Figure 7 shows the testing environment for basic RTIWS, and figure 8 is for extended RTIWS. PingPong test method is apopted in the experiment. In each experiments, two clients communicating with each other. Client No.1 sends update while client No.2 receives the data, then client No.2 immediately sends received data back to client No.1. Time-latency is half of the time between sending and receiving for client No.1.

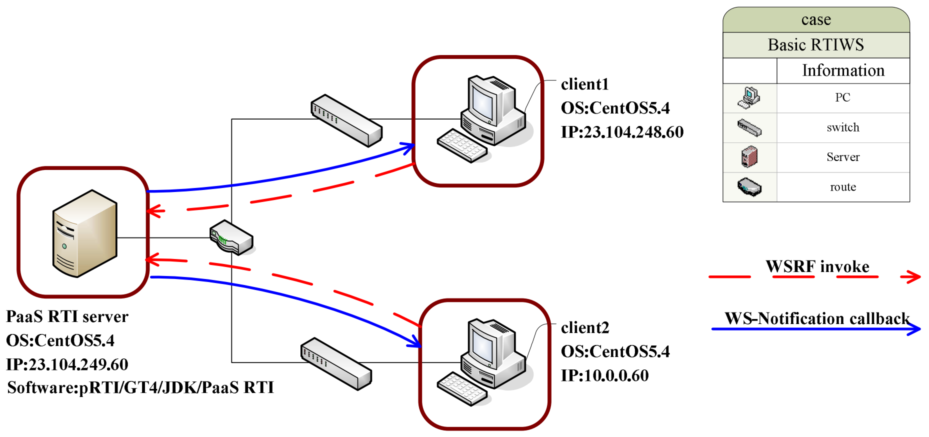

Fig. 7 Testing Environment for Basic RTIWS

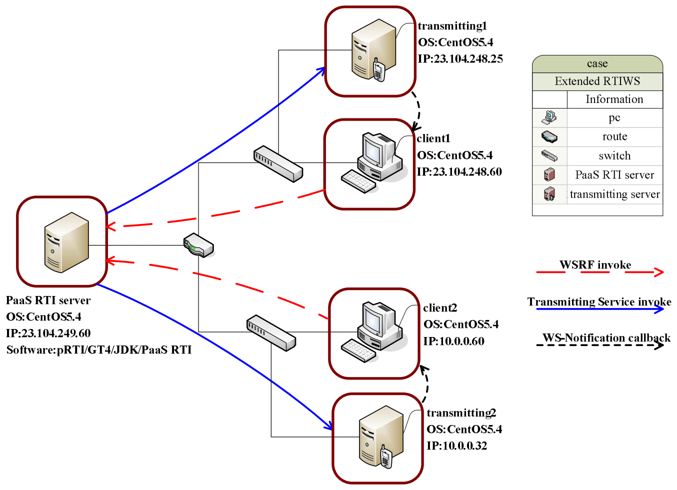

Fig. 8 Testing Environment for Extended RTIWS

Of course, different length of the update data would affect the time-latency in a certain degree. In the test, the length of the update data changes from 2 bytes to 10000 byte.

Figure 9 is the testing outcome for basic RTIWS, and figure 10 is for extended RTIWS. It is visible that the time-latency of RTIWS-based communication is longer than time-latency of communication directly through RTI, because some time is used to coding and decoding. We can also find that the time-latency of extended RTIWS is longer than basic RTIWS, because extended RTIWS sends callback information to client through the transmitting member, and the route is one more step than using basic RTI.

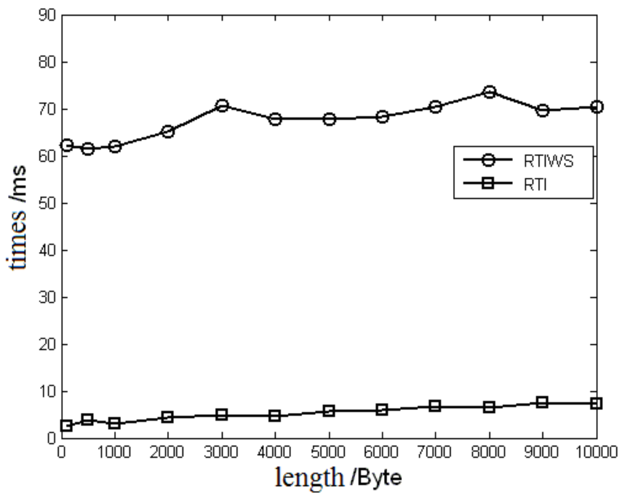

Fig. 9 Testing Environment for Extended RTIWS

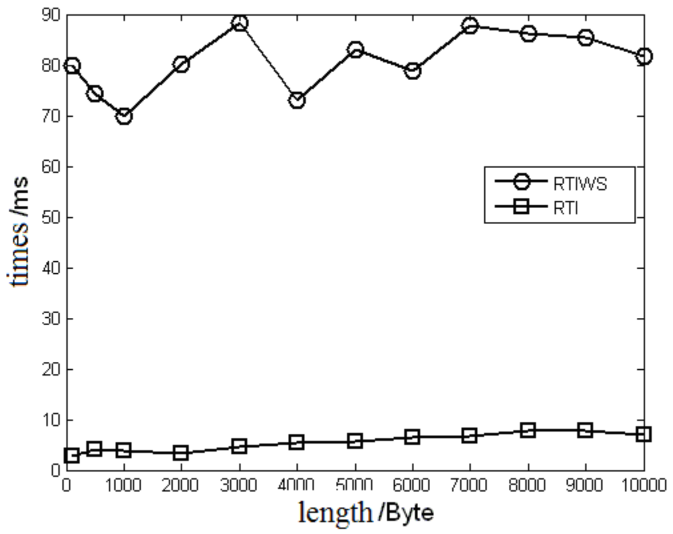

Fig. 10 Testing Environment for Extended RTIWS

Conclusions

In this paper, we propose PaaS RTI to supporting distributed simulation interaction in cloud computing age. PaaS RTI mixes the HLA/RTI and WSRF together. It is Web Service in essence, and provides reusability and flexibility to HLA/RTI development and course. RTIWS is core of PaaS RTI, and we put forward two types of RTIWS architecture, basic RTIWS is for simple LAN, and extended RTIWS is for common LAN.

References

[1] Ke Pan, Stephen John Turner, Wentong Cai, et al. A Service Oriented HLA RTI on the Grid[C]//In proceedings of 2007 IEEE International Conference on Web Services, 2007

[2] K. Rycerz, Grid-Based HLA Simulation Support[D].PhD thesis, University van Amsterdam and AGH Krakow, July 2006.

[3] Wentong CAI, Stephen J.TURNER, Hanfeng ZHAO. A Load Management System for Running HLA-based Distributed Simulations Over the Grid[C]//In Proceedings of the 6th IEEE International Workshop on Distributed Simulation and Real-Time Applications (DS-RT 02), 2002.

[4] Katarzyna Zajac, Marian Bubak, Maciej Malawski. Towards a Grid management system for HLA-based interactive[C]//Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT’03), 2003:4-11

[5] Liu Hong, Hu Wan, Yu Wang, et al. Extending HLA/RTI to WAN Based on Grid Service[C]//In proceedings of 2008 IEEE Aisa-Pacific Services Computing Conference, 2008:57-62.

[6] Hengye Zhu, Guangyao Li and Lulai Yuan. WSHLA: Web Services-Based HLA Collaborative Simulation Framework. Y. Luo (Ed.): CDVE 2007, LNCS 4674, 272-279

[7] G. Fox, A. Ho, S. Pallickara, et al Grids for the GiG and Real Time Simulations[C]//In Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Applications, 2005:129-138.

[8] K.Zajac, M.Bubak, M.Malawski, and P.M.A.Sloot. Towards a Grid management system for HLA-Based interactive simulations[C]//In Proceeding of 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications(DS-RT2003), 2003:4–11.