Dynamic Structure in Peer-to-Peer Based Distributed Simulation

Azzedine Boukerche, Ming Zhang, Hengheng Xie
Paradise Research Lab
School of Information Technology and Engineering (SITE)
University of Ottawa, 800 King Edwards St.
Ottawa, Ontario, K1N 6N5, Canada
{boukerch, mizhang, hxie072}@site.uottawa.ca

Keywords: Dynamic Structure, Distributed Simulation, Time Management Service

Abstract

Distributed simulation have been an dominant technique in recent years due to its capability in solving large-scale and complex simulation problems. Traditional distributed simulation techniques, such as HLA/RTI, are facing new challenges due to the fast advances in parallel and distributed computing techniques. Moreover, the increasing interests in flexible and scalable Peer-to-Peer (P2P) technique opens up new direction for us to discover the potentials for establishing next-generation distributed simulation standards and methodologies. In this paper, we propose a novel "dynamic structure" technique in the context of a P2P based distributed simulation infrastructure. This new technique aims to provide a highly reconfigurable distributed simulation environment, in which simulation entities can join/leave a running simulation at any time without significantly disturbing the core Time Management Service (TMS). Moreover, we implemented a prototype of this new technique, which demonstrated how a model can dynamically join a running simulation, automatically discover the TMS in the network, and then exchange messages with TMS and other running models, all in a autonomous fashion. We believe that our proposed "dynamic structure" technique can greatly promote the in-depth research for next-generation of distributed simulation, which needs to address more on reconfigurability, self-awareness, interoperability, and etc.

1. INTRODUCTION

With the increasing demands for distributed simulation, many research has been conducted in order to enhance existing distributed simulation frameworks or to propose novel solutions. In recent years, the importance of reconfigurable simulation, especially in distributed environment, is aware due to the fact that many of today's simulation problems need the simulation models to be dynamically reconfigurable. Such techniques have been addressed by many simulation researchers. For instance, [1] proposed a technique called "variable structure" in the context of Discrete Event System Specification (DEVS) [2], which is able to reconfigure the running simulation models based on predefined events in other model's specifications. As a widely recognized distributed simulation standard, mainly for interoperability of network distributed simulation entities, HLA [3] provides a rich set of services to address simulation time management, data management, simulation objects registration and management, and etc. However, it supports limited model run-time reconfigurations, and many recent research, in terms of HLA based simulation reconfigurations, are focused on the federates migrations which in most cases require "freezing" the time management service and checkpointing/restoring the migrated federates run-time states. As a matter of fact, the existing approaches for reconfigurable distributed simulation are largely confined by the traditional client/server based paradigm and the middleware based solutions.

In contrast, in this paper, we propose a novel solution for simulation reconfiguration in the context of P2P network environment, where new simulation entities can freely join/leave a running simulation without "freezing" the TMS as traditional approaches do. Our technique allows the newly involved simulation entities to pace themselves with TMS and other running simulation entities in an autonomous fashion. Moreover, the dynamic model structure changes can happen at any time during a simulation execution period, and such changes can be realized as a predefined event-triggered or as a "at-will" mechanisms. Indeed, our approach is based upon our previous work in Time Management Service (TMS) in a P2P based distributed simulation infrastructure [4], and this new technique aims to provide a highly reconfigurable TMS in a distributed simulation environment, which can support dynamic simulation model reconfiguration as well as seamless run-time model joining/leaving. Compared with traditional approaches, our approach does not need to "freeze" the TMS and checkpointing/restoring the run-time model states information, and therefore, the overhead generated by "freezing" TMS is greatly reduced. Furthermore, we implemented a prototype of this new technique, which demonstrated how a model can dynamically joining a running simulation, automatically discovering the TMS in the network, and then exchanging messages with TMS and other running models.

We will discuss the detail of our proposed techniques in the following sections, which are organized as follows: Sec-

tion 2 introduces related background with a focus on JXTA and some recent distributed simulation tools in terms of reconfigurability; Section 3 presents our proposed "dynamic structure" technique in JXTA based distributed simulations; Section 4 demonstrates a prototype example on dynamic reconfiguration of a running simulation; Section 5 concludes this paper with suggestion for some future work.

2. BACKGROUND

Peer-to-Peer (P2P) based networking technology is an emerging technique in recent years. Compared to traditional client/server or middleware based distributed system architectures, P2P system is able to provide higher level of flexibility, scalability and robustness. The key idea of P2P is to eliminate the bottleneck caused by a centralized server, and therefore, the overall system efficiency and resource utilization are improved. Indeed, many new distributed system concepts are now introduced based upon the new Peer-to-Peer philosophy. As one of the well-known P2P techniques, JXTA [5,6] becomes an reliable and efficient application framework, which supports rapidly developing advanced P2P applications. Basically, JXTA provides a rich set of protocols to support network peers that can discover, communicate, and monitor one with another. The key concept of JXTA is demonstrated in Figure 1, in which, we can see that a virtual network is built upon "peer groups" and their relationships. "Peers" can easily join or leave "Group" through an publish/discover mechanism supported by JXTA peer discovery protocol. JXTA has six solid built-in protocols to enable autonomous P2P networking, which includes: Peer Discovery Protocol (PDP), Peer Information Protocol (PIP), Peer Resolver Protocol (PRP), Pipe Binding Protocol (PBP), Endpoint Routing Protocol (ERP), and Rendezvous Protocol (RVP) [6]. Therefore, the applications built upon these protocols can take the advantages of such a highly reconfigurable and self-aware heterogeneous virtual computing infrastructure. It worth mentioning that JXTA uses Pipe based concept for message passing and XML format for common message encoding and service advertisement, which makes it capable of supporting interoperability, platform independency, and ubiquity [5].

As a matter of fact, P2P based concept has not yet been widely used in the distributed simulation area. In the contrary, many existing distributed simulation frameworks are still use middleware based solutions to extend single processor's simulation towards large-scale distributed environment. For instance, [8] implemented HLA's key functionalities as Grid service, thus, provided a distributed simulation framework based on invocations of Grid services. Other tools, such as DEVS/SOA [9], DCD++[10], used the same strategy to wrap their existing single processor's simulators as standard web services to form a integrated distributed simulation environment with the Internet as backbone. Meanwhile, ADEVS[11], PCD++[12] are DEVS based distributed simulation tools which are based on PVM and MPI. In term of reconfigurability and flexibility, these solutions are largely confined by the underlying non-dynamic architectures provided by HLA, MPI, and etc. Considering the limitation of these tools, some efforts have been given in order to achieve higher level or reconfigurability and robustness in a distributed simulation environment, which include DEVS/RMI [13] DEVS/GRID[14], DEVS/P2P [15]. However, these tools lack of interoperability and can only provide limited reconfigurability.

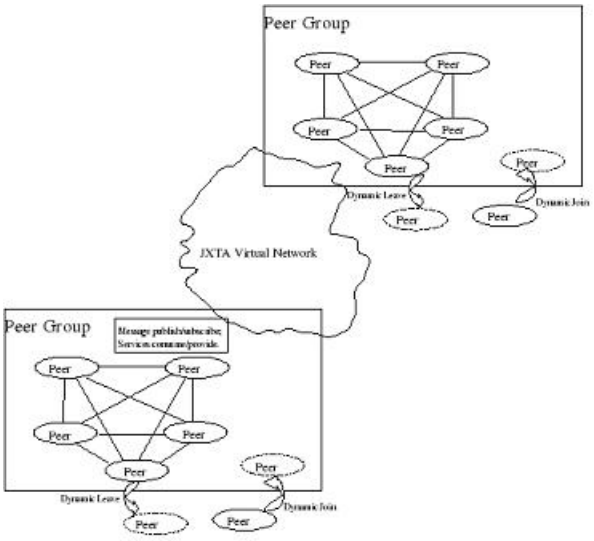

Figure 1. JXTA Peer Group Based Virtual Network [7]

3. DYNAMIC STRUCTURE IN JXTA BASED DISTRIBUTED SIMULATION

In this section, we will propose our idea on simulation entities run-time reconfiguration based on our previous work on distributed simulation framework using JXTA P2P technique. We first review our previous work on JXTA based Time Management Service (TMS), and then focus our discussion on how a "dynamic structure" is realized and implemented based upon JXTA's peer discover protocol, simulation entities' publish/subscribe advertisement mechanisms, and JXTA based simulation entities' capabilities for dynamic creations of input/output ports.

3.1. Time Management Service (TMS)

We will first briefly review the TMS that we proposed and implemented in our previous work demonstrated in [6]. The

TMS is in fact a JXTA group based service, which can automatically publish its existence and handles the time advance requests among all simulation entities within a group. It implements a conservative time management algorithm using JXTA Pipe concept, which makes it a robust and stateless core unit in the overall distributed simulation framework [anss-08]. The concept of "plug-and-play" is used to support high level of reusability and interoperability between simulation components. For instance, we can easily replace one time management implementation with another without affecting any other parts of the system as long as the replaced one conforms to the same input/output pipes' definitions. Taking the advantages of JXTA virtual network, our TMS can be transparently migrate to any computing node without affecting the overall system integrity. It worth mentioning that the TMS only handles the time advance and synchronization of all involved simulation entities, while the message passing or event propagation happens directly among the entities. Thus, the TMS is lightweight and stateless, which makes it easier to support dynamic structure changes in the simulation system. Moreover, the TMS and the simulation entities and can also interact directly with other simulation services whenever necessary. As we can see, such approach makes the deployment and partition of simulation entities much easier compared to those middleware based solutions.

3.2. Dynamic Structure

Based upon our previous research work on TMS, we take a step further on proposing and implementing a "Dynamic Structure" mechanism that can significantly enhance the flexibility and adaptiveness of our distributed simulation runtime management. Differentiating it from existing approaches for reconfigurable distributed simulation frameworks, our approach aims to provide a higher level of reconfigurability, self-awareness, robustness.

The key idea of our approach is that the TMS is lightweight, stateless and self-adaptive. Thus, whenever new simulation entities join a running simulation, TMS can automatically adjust itself to the changes in terms of system structure. The TMS actually has a designated input/output pipes to handles the dynamic entities' registration (pipe event based mechanism), and then automatically paces itself to the system structure's change before entering next simulation loop. Therefore, the overall simulation execution is non-interrupted, and the new simulation entities then join the running execution seamlessly.

As shown on Figure 2, the dynamic simulation entities lookup the TMS service during a simulation execution, then register themselves to the TMS; TMS then replies with "registration done" message back, based upon which the dynamic entities create necessary input/output pipes during run-time, then automatically pace themselves to the forwarding simulation loops. In other words, the overall distributed simulation is a self-awareness and self-adaptive system, in which, simulation entities and the TMS can automatically discover each other and adapt to any changes in the system, all in an autonomous fashion. It worth mentioning that our proposed "dynamic structure" technique is based on the most advanced JXTA service protocols, which makes it different from traditional approaches that rely on complex procedures of state checkpointing, "freezing" the TMS, and etc. In the following section, we will present our prototype based experiment which implements the idea we demonstrated in this section.

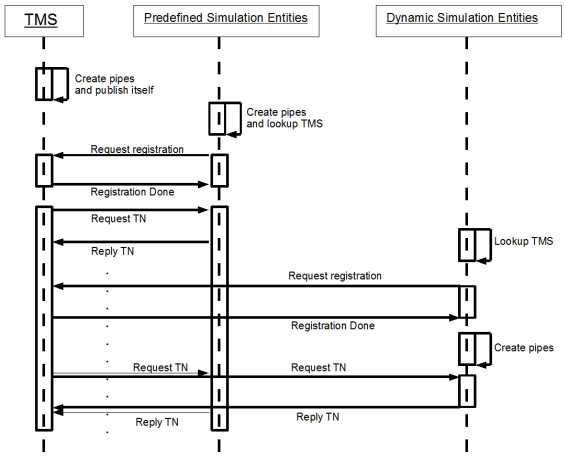

Figure 2. Sequence Diagram for Dynamic Simulation Entity Joining

4. EXPERIMENT

Our experiment is conducted using two machines and four individual JXTA-enabled Java processes. One machine (bouker19) is equipped with Intel Pentium 4 3.2G CPU with 1G memory, and another machine (bouker5) has a Intel Pentium 4 3.0G CPU with 512M memory; they are in the same 100M sub-network. Both machines are installed with Linux 2.6.20, JDK 1.6.0_04 and JXTA library version 2.5_rc1.

In the following experiment, we will use two types of simulation entities and JXTA service based implementation of TMS. Followings are a briefly descriptions of involved simulation entities and TMS:

Generator: a simple simulation entity that periodically generate "task" message to other simulation entities. The time of next event of a Generator is period*n, where period is a predefined model parameter as double value, and n=1,2,3,...

Processor: a simple simulation entity that process received "task" using a predefined interval. The time of next

event of a Processor is event_time+processing_time*n, where event_time is the clock time when Processor receives a "task", processing_time is a predefined model parameter as double value, and n=1,2,3,...

TMS: a JXTA service that implements a conservative time management algorithm as described in section 3.1. TMS has input/output pipes for simulation entities registration, message exchanging, and a specified pipes for handling dynamic simulation entities registration.

In this experiment, we first simulate a generator-processor (g-p) model, and a generator-processor-processor (g-p-p) model respectively using one machine(bouker19). As shown in Figure 3, the TMS starts on one console window first, then "generator" and "processor" entities start in the other two consoles windows. The sequence of starting these three processes does not matter because the "generator" and "processor" entities all use the infinite loop to search the TMS in the network, and then register themselves to the TMS. The main simulation control loop is handled by the TMS after all involved simulation entities have been registered. As shown in Figure 3, the TMS has an output pattern as "3, 5, 8, 10, 13, 15, 18, 20,..." , which is actually the time of next event among all models. Similarly, Figure 4 is the dump screen of running g-p-p model, where another "processor" entity is started from another machine (bouker5). In this case, the TMS is set to accept three simulation entities' registration before the simulation loop starts. The TMS now has output pattern as "3, 4, 5, 8, 9, 10, 13, 14, 15, 18,19 20, ..." because the second "processor" initiated on bouker5 has a processing time of 4 as model internal parameter.

Now we will see how we can reconfigure a "g-p" model to a "g-p-p" model during the simulation run-time. As shown in Figure 5, the TMS is initially set to accept two simulation entities' registration before the simulation loop starts. Therefore, at first, the TMS starts its loops with a output pattern the same as the "g-p" model shown in Figure 3. During this simulation execution, we starts the second "processor" on machine "bouker5" at any time "at will", the TMS is then automatically adapt itself to this system structure changes, and then simulate the "g-p-p" model thereafter. The output pattern of TMS then conforms to the pattern shown in Figure 4. As a matter of fact, when the second "processor" starts on "bouker5", it sends out a registration message to the running TMS, which then uses its dynamic registration pipe to receive this registration event. TMS then updates its registered entities to 3, and sends out a message to this newly joined "processor" for creating necessary pipes before TMS enters to its next cycle. All this happens seamlessly in TMS without "freeze" and "resume" mechanism. In other words, the TMS is self-awareness of the dynamic model structure changes and also self-adaptive to this change in a totally autonomous fashion.

As a matter of fact, all these Java processes used in this experiment can be placed on any selected machines in a local sub-network to generate same result without any changes of the codes. This is due to the JXTA's powerful set of peer based protocols.

5. CONCLUSION AND FUTURE WORK

In this paper, we extends our previous research work on P2P based distributed simulation infrastructure with a novel technique called "dynamic structure", which seamlessly integrate itself to our existing JXTA based Time Management Service (TMS). We demonstrate a prototype implementation of this new technique that verifies the feasibility of using JXTA based service protocols for developing next generation distributed simulation infrastructure, in which higher level of flexibility, robustness, and self-awareness can be easily addressed. Our proposed simulation run-time reconfiguration technique can be used for distributed interactive simulation, gaming based simulation, as well as traditional analytical based simulation, in which users(or new simulation entities) can join a running simulation at any time "at will". Compared with existing approaches, our approach does not need to "freeze" the TMS and checkpointing/restoring the run-time model states information, and moreover, the system's model structure changes are not necessary predefined. We believe that our proposed "dynamic structure" technique, in the context of JXTA based distributed simulation infrastructure, can greatly promote the in-depth research in next generation distributed simulation, which not only focus on high performance, but also higher level of interoperability, reconfigurability and self-awareness.

For the future work, we will further standardize our approach, and also apply it towards complex simulation problems. The overhead and performance in a distributed network will be measured to obtain the clues for further optimizing our approach.

6. REFERENCES

[1]. X. Hu, B.P. Zeigler, "Model Continuity in the Design of Dynamic Distributed Real-Time Systems", IEEE Transactions On Systems, Man And Cybernetics— Part A: Systems And Humans, 35(6), 2005, pages 867- 878.

[2]. IEEE Standard. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules. 1516-2000, September 2000.

[3]. B. P. Zeigler, T.G. Kim, and H. Praehofer, "Theory of Modeling and Simulation", 2 ed., New York, NY: Academic Press, 2000.

[4]. Azzedine Boukerche, Ming Zhang, Hengheng Xie, "An Efficient Time Management Scheme for Large-Scale Distributed Simulation Based on JXTA Peer-to-Peer Net-

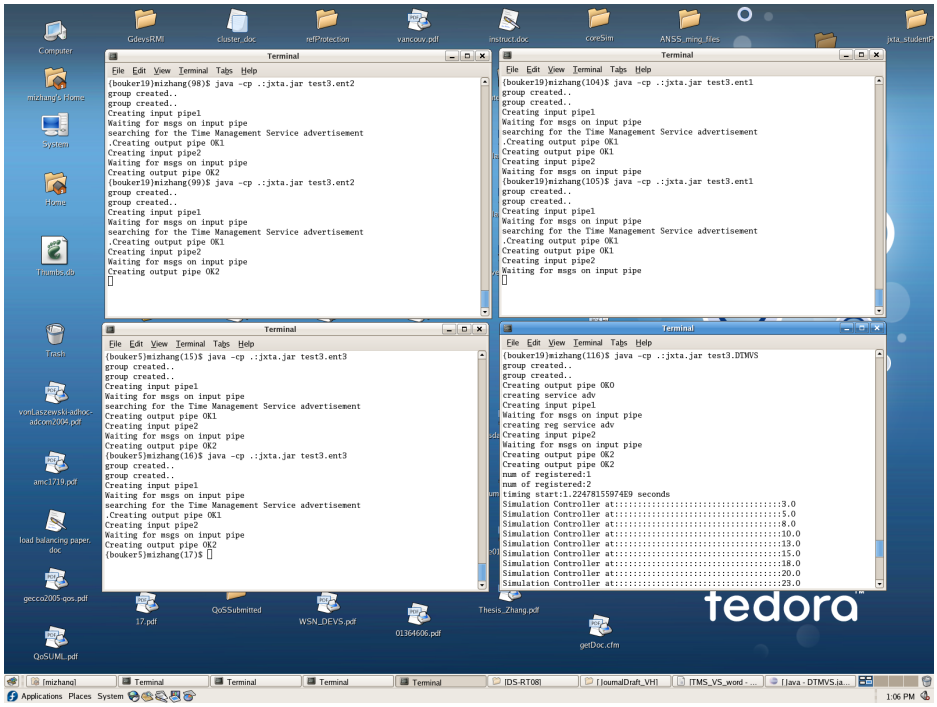

Figure 3. Running g-p model

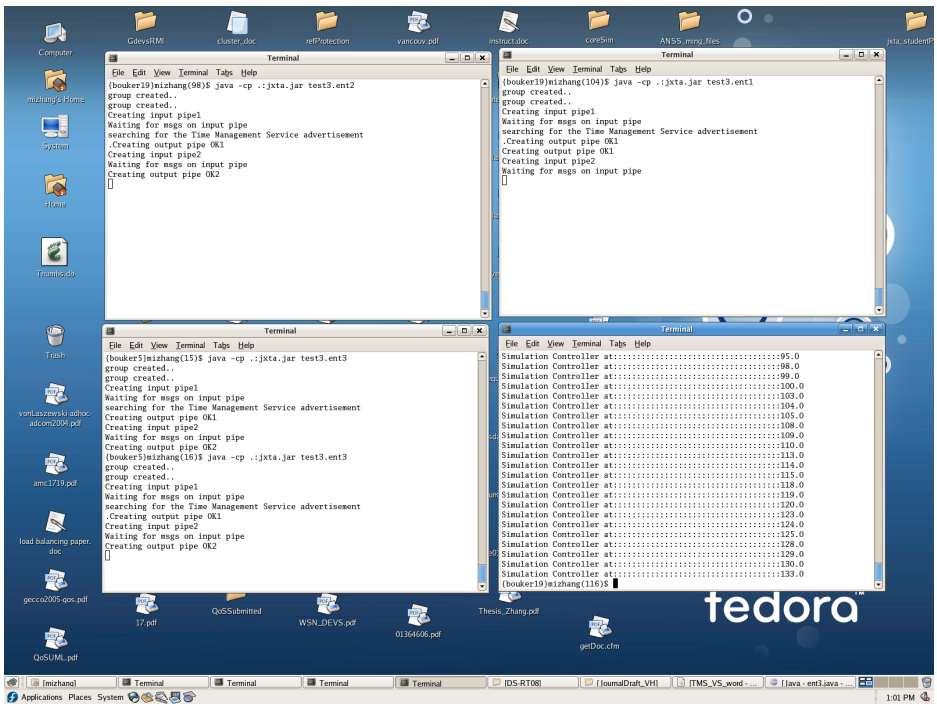

Figure 4. Running g-p-p model

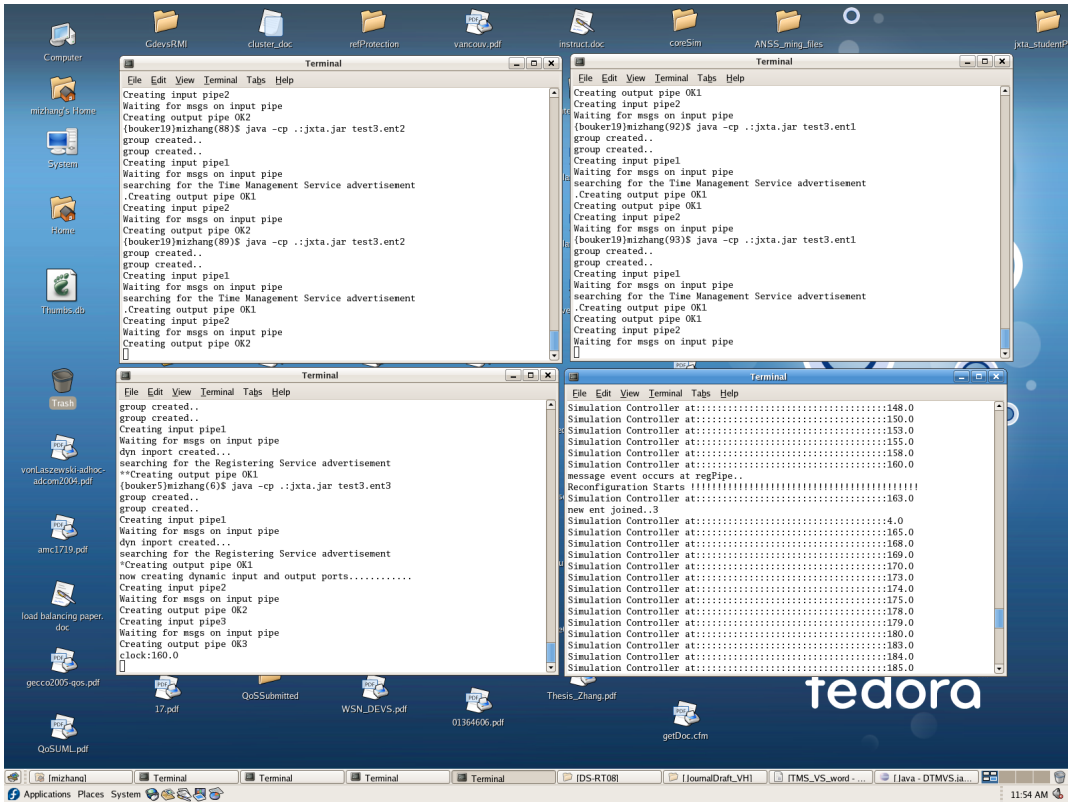

Figure 5. Reconfiguring the g-p model to g-p-p model in run-time

work", Proceedings of IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, 2008.

[5]. JXSE 2.5 Programmers Guide: JXTA Concepts, online at: https://jxta-guide.dev.java.net/source/browse/*checkout*/jxta-guide/trunk/src/guide_v2.5/JXSE_ProgGuide_v2.5.pdf

[6]. JXTA Protocols Specification, online at http://jxta-spec.dev.java.net.

[7]. A. Boukerche,M. Zhang,"Towards Peer-to-Peer Based Distributed Simulations on a Grid infrastructure", 41st Annual Simulation Symposium(ANSS-41 2008),pp. 212-219.

[8] K. Pan, S. J.Turner, W. Cai , Z. Li, "A Service Oriented HLA RTI on the Grid", 2007 IEEE International Conference on Web Services (ICWS 2007).

[9]. S. Mittal, J. L. Risco-Martin, B.P. Zeigler, "DEVS-Based Simulation Web Services for Netcentric T&E", Summer Computer Simulation Conference SCSC'07, July 2007.

[10]. J. Chazal, L. Quinet, Q. Liu, M. K. Traoré, G. Wainer, "Performance Analysis of Web-based Distributed Simulation in DCD++: A Case Study across the Atlantic Ocean", HPCS, part of Spring Simulation Multiconference 2008.

[11]. ADEVS, http://www.ornl.gov/~1qn/adevs/index.html

[12] Q. Liu, G. Wainer, "Parallel Environment for DEVS and Cell-DEVS Models", Simulation, Transactions of the SCS. Vol. 83, No. 6, 449-471 (2007).

[13] Ming Zhang, B.P. Zeigler, P. Hammonds, "DEVS/RMI-An Auto-Adaptive and Reconfigurable Distributed Simulation Environment for Engineering Studies", International Test & Evaluation Association (ITEA) Journal of Test and Evaluation, March/April 2006, Volume 27, Number 1, Page 49-60.

[14]. C. Seo, S. Park, B. Kim, S. Cheon, B. P. Zeigler, "Implementation of Distributed High-performance DEVS Simulation Framework in the Grid Computing Environment", 2004 High Performance Computing Symposium.

[15]. S. Cheon, C. Seo, S. Park, B. P. Zeigler, "Design and Implementation of Distributed DEVS Simulation in a Peer to Peer Network System", 2004 Military, Government, and Aerospace Simulation.