Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

A NOVEL MESSAGE-ORIENTED AND SOA BASED REAL-TIME MODELING AND SIMULATION FRAMEWORK FOR PEER-TO-PEER SYSTEMS[1]

Hengheng Xie
Azzedine Boukerche
Ming Zhang

Paradise Research Lab
University of Ottawa
Ottawa, ON, K1N 6N5, Canada
{ hxie072, boukerch, mizhang} @site.uottawa.ca

ABSTRACT

Recent advances in Service Oriented Architecture (SOA) provides many exciting opportunities for developing next-generation of distributed simulation frameworks and tools. At the mean time, Peer-to-Peer (P2P) based network technique also challenges the traditional view of distributed simulations. Indeed, the integration of SOA and P2P techniques can potentially help on developing more flexible, scalable distributed simulation framework. In this paper, we present our design and implementation of a real-time distributed simulation framework based on SOA concept and JXTA P2P technique. Our simulation framework can be effectively used for evaluating most of SOA related algorithms and schema including but not limited to: dynamic service composition, service path selection, load-balancing algorithms, and etc. Meanwhile our framework can also be applied to emergency preparedness class of applications to identify the critical parameters for designing more efficient emergency response systems.

1 INTRODUCTION

Service Oriented Architecture (SOA) (Erl 2005) rapidly becomes a dominant approach for system development and integration, which allows different software service components to exchange information through various networks. SOA is based on the concept of "services", which defines some software units that use pre-defined protocols to communicate with each other. In service-oriented system, services expose themselves using interfaces, and service consumers access the services through these interfaces without the awareness of the detailed service implementation. The main advantages of SOA are flexibility, interoperability and reusability. Indeed, SOA is a promising technique, which has attracted many developers to implement their key software backbones using SOA architecture.

As a matter of fact, more and more researchers dedicate themselves on discovering the new features of SOA. However, one of the biggest concerns is how to effectively design a SOA based system considering its design complexity. With the increasing usage of services in a SOA based system, the system designers may be overwhelmed due to large number of service components. On the other hand, in order to satisfy the requirements such as security and reliability, more functionalities need to be added to the original service based system. For instance, the description of recent business processes is becoming more and more complicated, which requires the designers to consider more on how to initialize and maintain the interoperations of the whole system.

In this paper, we propose a novel service oriented real-time modeling and simulation framework which aims at solving design problems of SOA based systems. We implement our framework using Sun's JXTA P2P API. Our framework is built on JXTA's service backend, thus, it can easily and accurately modeling and simulating many of the real world and real-time SOA based systems, especially P2P based system architectures. Moreover, our framework is based on message layers and P2P based services, which makes it suitable for simulating SOA based systems in a real-time fashion. Because our framework is built upon existing JXTA API, the services in our framework can easily communicate with each other using JXTA

[1] This work is partially supported by Canada Research Chair Program, NSERC, Ontario Distinguished Researcher Award, Ontario Early Researcher Award, EAR Award and ORF Funds.

protocols. Meanwhile, a set of services can be organized as a group in our framework, which reduces the complexity of the interoperation of services. Taking the advantage of the discovery protocol of JXTA, services in our framework can publish or discover each other efficiently across network. Furthermore, services residing at different subnets can communicate with each other using the super, rendezvous and relay peers. Inheriting these key features of JXTA, it's very easy to model and simulate a service-oriented system using our framework.

The rest of the paper is organized as following: Section 2 presents the related works for SOA based modeling and simulation approaches; Section 3 proposes our design; Section 4 demonstrates how we implement our framework using JXTA; Section 5 describes two related experiments; and finally Section 6 concludes the paper with some suggested future work.

2 RELATED WORKS

Service Oriented Architecture (SOA) has become an emerging technique in recent years. Thus, many novel techniques have been proposed for improving the functional or non-functional features of SOA. However, due to the increasing complexity of service-oriented systems, it is generally difficult to design and organize such systems. In terms of modeling SOA based system, many researchers proposed new techniques and methods (Zeigler, Kim and Praehofer 2000). Normally, these techniques can be classified as functional modeling and non-functional modeling. The purpose of functional modeling is to describe the functions and processes of the system, which helps to obtain the necessary information and to identify the basic function of the system. For instance, Sloane proposed a hybrid approach for modeling SOA systems using Colored Petri Nets (CPN) and MESA/Extend (Sloane et al. 2007), which model the SOA system in two different layers: component level and black-box level. These two levels do not affect each other and can be validated separately. Similarly, Kogekar presented a Middleware Building Block approach for SOA system modeling and performance analysis using Stochastic Reward Net (SRN) (Kogekar et al. 2006). SRN is very close to designer's intuition and is very helpful for understanding the created model. Model Driven Development approach(Vale and Hammoudi 2008) has also been used for modeling SOA based system. For instance, Vale and Hammoudi presented a Context-aware Service Oriented Architecture based on such approach. In terms of modeling web services, Krause demonstrated an approach to integrate existing system with intelligent models(Krause et al. 2007). They used semantics modeling ontology to describe web services.

On the other hand, non-functional modeling tries to illustrate the specified criteria that is used to judge the operation of the system without focusing on its behaviors. For instance, Quality of Service (QoS) is one kind of non-functional requirements, which intends to provide guaranteed performance to users. With regard to the research for modeling the non-functional aspects of SOA systems, Wada proposed a non-functional modeling approach with UML profile (Wada, Suzuki and Oba 2006), which can specify and maintain the SOA's non-functional features in an implementation independent manner. They also presented another new Model Driven Development (MDD) framework to model the non-functional constrains of SOA system(Wada, Suzuki and Oba 2007). Similarly, Zhiang and Junzhou proposed a QoS-Resource Graph Model (QRGM) for modeling end-to-end QoS (Zhiang and Junzhou 2007).

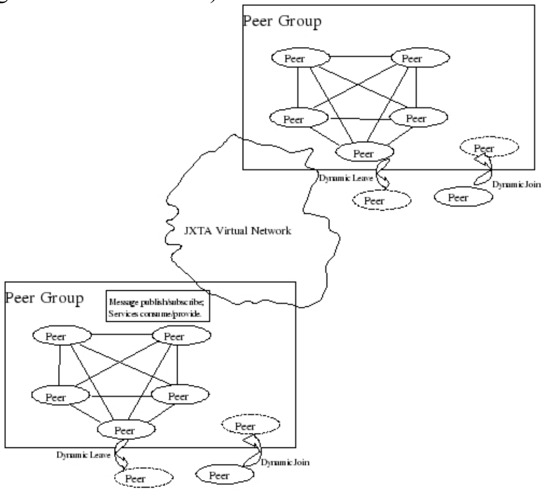


Figure 1: JXTA Virtual Network (Boukerche and Ming 2008)

It worth noting that the above surveyed approaches all focused on modeling SOA systems, however, they cannot be used to directly simulating the modeled systems. In this paper, we propose a novel SOA based simulation framework which is built upon JXTA overlay network and can effectively model and simulate real SOA implementations. We build a Message Layer on top of our P2P services framework to support high level SOA modeling and simulation. The virtual network built on JXTA in our framework is shown on Figure 1, in which, a set of JXTA peers can be organized as group and then communicate with each other within the same group. Peers in different groups can exchange messages only through some special peers.

3 FRAMEWORK DESIGN

In this paper, we design and implement a message-oriented and real-time SOA based modeling and simulation framework on top of JXTA P2P protocols. In our framework, all services and components are implemented as JXTA entities, which are able to create JXTA pipes to communicate with each other. Services can publish advertisements on the network, and they can be discovered by other services easily through JXTA discovery protocol. For the JXTA entities residing on different sub-networks, we use super, rendezvous and relay peers to support inter-subnet message communication. In an ordinary JXTA network, entities can initialize connections with other entities after discovering their advertisements. However, each JXTA pipe is maintained as an Java object in the memory. When the number of JXTA entities and connections increases, the usage of memory can increase dramatically at the same time.

In order to solve this problem, we create a Message Layer in our framework, as shown in Figure 2. This message layer is responsible for managing the whole message exchanging process, including pipe creation, messages sending and messages receiving. In particular, our approach separates the process of advertisement's discovery from pipe initialization. In the pipe initialization process, entities do not create any real pipes. If an entity need to send a message, it first discovers the advertisement of its destination, and then send the message to the Message Layer. When the Message Layer receives any messages, it redirects them to the destination. In other words, all messages in our framework use Message Layers for exchanging messages. Meanwhile, we also implement a Component Platform, which works as a monitoring and organizing manager for each node.

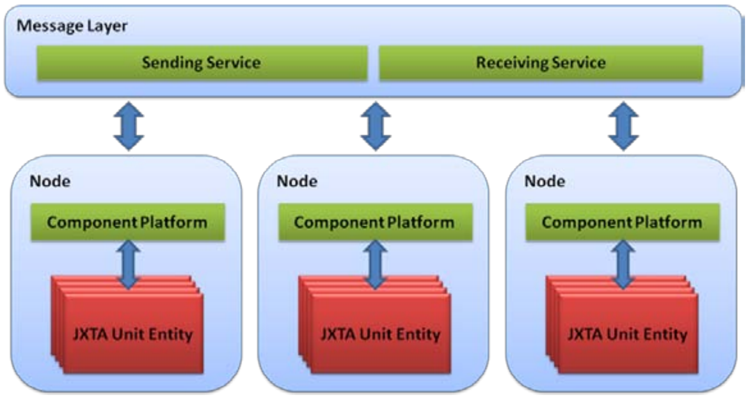


Figure 2: Framework design for SOA on JXTA

The advantages of our system structure are: 1) It reduces the number of pipe objects that need to be maintained. In many cases, part of the pipe objects may not be used frequently in the system, but they still exist in memory, which results in a waste of system resource. Thus, decreasing the number of pipe objects can save the system resource ; 2) In our approach, we simplifies the process for component migration when necessary. In SOA systems, components may need to be moved from one node to another node during a load balancing or fault tolerant process. In our solution, components do not need to be aware of the migration of other components. All components can still work as before after the migration and they do not need to establish the connections again; and 3) We also simplify the implementation of reliable JXTA pipes. In our solution, we can implement the reliable features of JXTA pipes on the Message Layer, without implementing them on all services used in the system.

4 SYSTEM IMPLEMENTATION

In our framework, we try to best utilize existing features of JXTA, such as the concept of peer group and peer advertisement, pipe and service discovery protocols, and etc. All these features can make the applications more flexible and configurable. However, we still have to do some improvements in order to implement our design presented in the previous section.

As we know, JXTA P2P network is a virtual overly network. In order to use the JXTA overlay network, an application needs to start a JXTA virtual network at first. It also need to create a global netPeerGroup object. In some cases, there exists a set of services running on one node, and they do not modify the netPeerGroup object. Usually, these services only create new peer group under the netPeerGroup object. Our idea is to share one netPeerGroup object among all the applications on the same node. As a result, the Component Platform is implemented as a process in order to fulfill this idea. Moreover, in our approach, the JXTA entities on the same node are implemented as threads under the Component Platform, which all shares one netPeerGroup object. Another advantage of our implementation is that all JXTA entities share the same cache folder, which is used to save the discovered advertisements for future usage. When one JXTA entity finds an advertisement, it saves it in the cache folder. Due to the cache folder sharing mechanism, other services can access the same advertisement, which can significantly save the time for searching advertisements.

Another problem that we have to solve in the implementation is that the Message Layer doubles the message that needs to be sent. In our design, each unit entity needs to send the message to Message Layer first, and then Message Layer will re-direct it to specified unit entity. This procedure makes each message goes through two routes: One from unit entity to Message Layer and another from Message Layer to unit entity. To solve this problem, we improve the Message Layer to a cooperated Message Layer on each node, as shown on Figure 3. In such an approach, there is a Message Layer running on each node, and all these Message Layer components can connect to each other to form a larger Message Layer across the network. JXTA entities work the same way as we described previously in section 3. However, the JXTA entities residing in the same node do not send message to Message Layer through JXTA pipes, instead, they directly use method call to send message to the Message Layer which is much faster. In other words, if two entities are running on the same node, their communication is totally manipulated by local method call without using any JXTA message. Thus, by using the cooperated Message Layer, we significantly reduce potential communication overhead among JXTA entities.

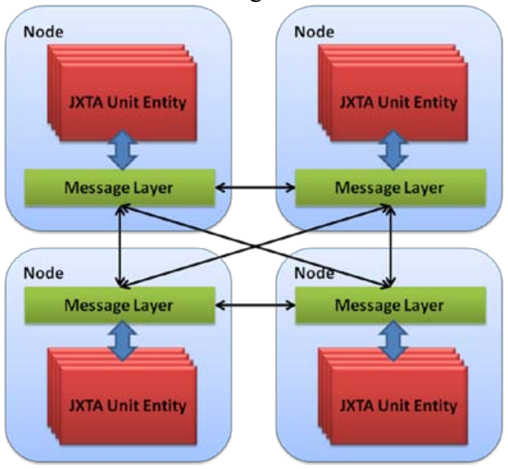


Figure 3: Improved cooperated Message Layer structure

In order to separate the discovery process from the pipe initialization process, we did some improvement on the discovery process. The real JXTA pipes only exist among Message Layers, so we do not need to create any JXTA pipe in JXTA entities. However, in the service advertisement of JXTA, it needs to contain the pipe advertisement for other entities in order to create the connection. In our framework, each entity does not contain any pipe information any more. We still want to use the discovery process that JXTA supplies, therefore, we have to do some improvements when encapsulating the advertisement. When creating the service advertisement, we do not embed the pipe advertisement of entities, instead, we put the pipe advertisement of Message Layer, which represents the entity for the message exchange. For example, when entity A discovers the service advertisement of entity B, it can retrieve the pipe advertisement of entity B's Message Layer from the service advertisement. Afterwards, entity A sends the pipe information to its Message Layer to create connection to entity B's Message Layer. Now entity A is able to send message to entity B.

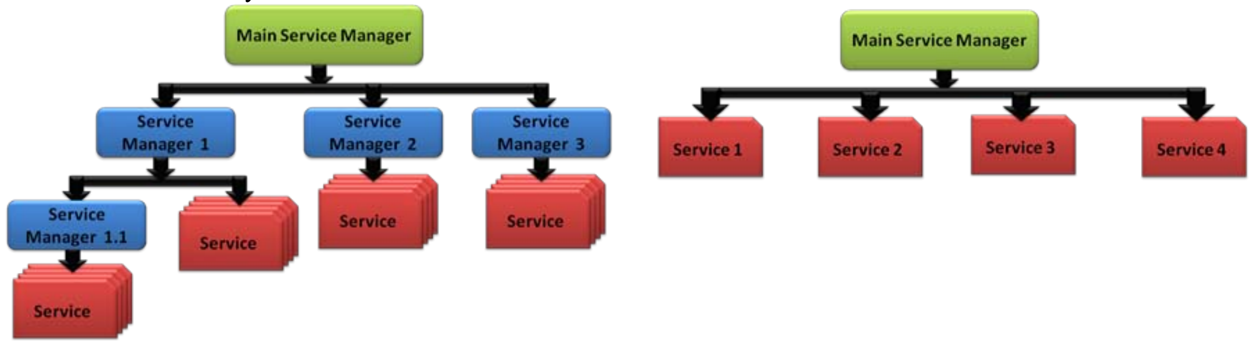
It worth mentioning that our solution does not affect other features of JXTA network. User is still able to create peer group to confine the communication between entities and to organize the overall system structure. Although all Message Layers are on the same peer group (which means there are no communication limit on the Message Layer), the process of advertisement discovery is still under control by peer group policy. That's also one of the reasons that we want to separate the communication process and the advertisement discovery process. As an example, Entity A in Group 1 wants to talk with Entity B in Group 2. Technically, Entity A is able to connect to Entity B through the Message Layer. However, Entity A will never find the service advertisement of Entity B because we still keep the discovery protocol of JXTA, which prevents such situation.

5 EXPERIMENTS

We conducted some experiments on our SOA based simulation framework. In this section, we present two of them to illustrate how to model and simulate SOA system using our framework. The first experiment aids our design of QoS-aware system on P2P network (Xie et al. 2008). The second experiment evaluates a load balancing algorithm for the system that we designed in the first experiment.

5.1 Simulating a QoS-aware Service Composition and Management System

In this experiment, we implemented a QoS-aware system on our SOA framework. The architecture of the QoS-aware system is shown in Figure 4(a). We compare our design to the flat service structure shown in Figure 4(b), which is widely used in service composition system. We implement and simulate both structures using our SOA based framework to compare the performance of these two systems.



(a) (b)

Figure 4: (a) Architecture for QoS-aware system (b) Architecture for flat service structure (Xie et al. 2008)

In our design, there are two main components in the system, manager and service. Both of them are implemented as JXTA entities. The basic functions of manager are to manage the services, to assign tasks and to evaluate the performance of the services. The Main Service Manager is the main manager that receives the tasks from the user and makes the final decision to assign the task to the services. Other Service Managers are controlled by the managers on the upper level. In this system, services are the unit functions of the system, and each service can finish one kind of unit tasks. The Service Mangers need to compose a set of services to finish certain task that is submitted by users. Each Service Manager only knows the components directly connected to it. When the Main Service Manager receives a task from a user, it forwards it to the Service Managers that it controls. The Service Manger that receives task from the higher level will propose a plan of service composition for part of the task. Then, it eliminates that part of the task and send it to the Service Manage on low level that it controls. After the whole task has been assigned, the system will start the execution process from the bottom to the top, and the final result is generated at the Main Service Manager. As a matter of fact, the Service Managers that do not depend on each other work in parallel. In contrast, in the Flat Service structure, only one Main Service Manager handles the service composition as well as monitoring the underlying services.

It's straightforward to compare above two system architectures using our simulation framework as shown in Figure 5. On the virtual communication layer that is built by the Message Layer, each component is able to connect to the components that it discovers. Components do not need to know the node on which the destination component runs. We can see from Figure 5, the Main Service Manager runs on a separated node because it needs more resource to process the tasks. As a matter of fact, we can run the service managers, services in whatever fashion we want in a cluster of computers. Our real-time simulation experiments demonstrated that our proposed architecture shown in Figure 4(a) had a better performance compared to the architecture in Figure 4(b).

5.2 Simulating a Load Balancing Scheme Based on Genetic Algorithm

In this experiment, we simulated a load balancing scheme for the QoS-aware system that we described in section 5.1. This load balancing scheme include two parts: a dynamic task scheduling algorithm and a service migration algorithm.

In the dynamic task scheduling algorithm, we create certain dynamic QoS properties for monitoring and estimating the performance of the services. Based on this QoS property, the service manager makes the decision for assigning the tasks. In the service migration algorithm, there is a load balancing manager monitoring the whole system that it controls. It probes every services and gets the big picture of the whole system. Then, it uses Genetic Algorithm to generate one of the best arrangement for the system. After getting the final blueprint of the system, service manager starts the service migration in order to get the final arrangement created by Genetic Algorithm. It worth noting that we also implemented a user monitoring service in our simulation framework for monitoring the system across the subnets.

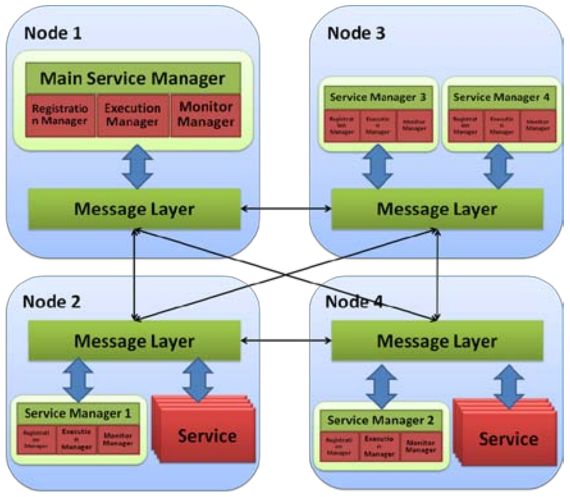


Figure 5: Implementation of QoS-aware system on the SOA framework

The system structure is almost the same as shown in Figure 5. However, we implemented some additional components in order to add the new scheduling function and service migration function. Due to the unique features of our framework, the service manager and services do not need to establish any connection to the migration service. The migration service only needs to communicate with the service manager when it starts and finishes the migration. We also evaluated the overhead of the service migration through simulation in our framework, and found that the overhead is within an acceptable level. In fact, the services that are built on our framework are able to migrate easily and efficiently from one node to another, which is very useful to simulate and evaluate various service composition schema and load-balancing algorithms.

Moreover, we also created some JXTA super peers to support the communication across network. As shown in Figure 6, Super Peer runs on both sub-networks that need to communicate with each other. The involved JXTA entities have to register to the super peer in order to publish and discover the advertisements to other subnets. We have also conducted some simulation experiments based on such scenario.

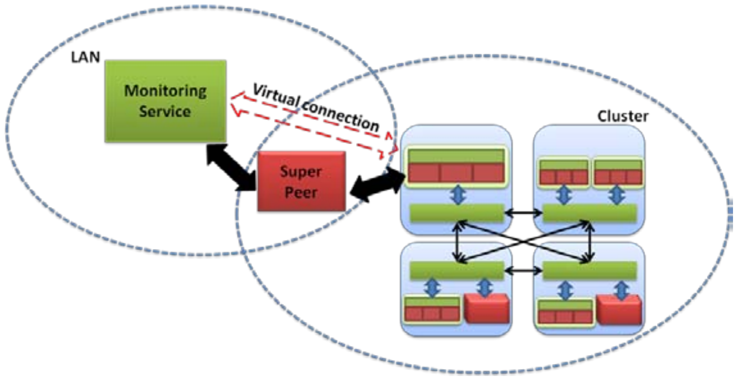


Figure 6: Super peer creates virtual connection between two subnets

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a novel message-oriented and SOA based real-time modeling and simulation framework focusing on simulating SOA and P2P based systems. We present the detail of our design and implementation to illustrate how our framework is suitable for real-time simulation of SOA based system architectures. We also introduce some related simulation experiments we have conducted to demonstrate the effectiveness of our framework on simulating and evaluating SOA based systems. The experiments demonstrate that our framework can support high-level of modeling and simulating SOA systems, with which, designer can focus on the workflow of the services as well as the service composition algorithms. Furthermore, it is transparent to realize a simulated SOA architecture to a real one because our framework itself is built upon distributed real-time SOA architecture.

For the future work, we plan to integrate the web service into our framework in order to expand the coverage area of our framework. We are also interested in integrating this framework with the P2P time management scheme (Boukerche, Zhang and Xie 2008) to establish distributed simulation across different sub-networks.

REFERENCES

Boukerche, A. and M. Zhang. 2008. Towards Peer-to-Peer Based Distributed Simulations on a Grid Infrastructure. In *Proceedings of Simulation Symposium, 2008. ANSS 2008. 41st Annual*. 212 – 219. Ottawa, ON, Canada.

Boukerche, A., M. Zhang, and H. Xie. 2008. An Efficient Time Management Scheme for Large-Scale Distributed Simulation Based on JXTA Peer-to-Peer Network. *In Proceedings of The 12-th IEEE International Symposium on Distributed Simulation and Real Time Applications*. 167-172. Vancouver, BC, Canada.

Erl, T. 2005. *Service-Oriented Architecture: Concepts, Technology, and Design*. Indianapolis, IN, USA: Prentice Hall PTR

JXSE 2.5 Programmers Guide: JXTA Concepts, Available via https://jxtaguide.dev.java.net/source/browse/*checkout*/jxtaguide/trunk/src/guide_v2.5/JXSE_ProgGuide_v2.5.pdf [accessed May 2009]

JXTA Protocols Specification, Available via <http://jxta-spec.dev.java.net> [accessed May 2009]

Kogekar, A., D. Kaul, A. Gokhale, P. Vandal, U. Praphamontripong, S. Gokhale, J. Zhang, Y. Lin and J. Gray. 2006. Model-driven generative techniques for scalable performability analysis of distributed systems. In *Proceeding of 20th Parallel and Distributed Processing Symposium*. Rhodes Island, Greece.

Krause, L.S., L.A. Lehman, B.R. McQueary, A.P. Stirtzinger and S.A. Stirtzinger. 2007. The Role of Intelligent Models in the Implementation of Service Oriented Architectures. In *Proceeding of International Conference on Integration of Knowledge Intensive Multi-Agent Systems*. 63 – 68. Waltham, Massachusetts, England.

Sloane, E., T. Way, V. Gehlot, R. Beck, J. Solderitch and E. Dziembowski. 2007. A Hybrid Approach to Modeling SOA Systems of Systems Using CPN and MESA/Extend. In *Proceeding of 1st Systems Conference*, ed. Mo Jamshidi, 1 – 7, Hawaii, USA

Vale, S. and S. Hammoudi, 2008. Model Driven Development of Context-aware Service Oriented Architecture. In *Proceeding of 11th IEEE International Conference on Computational Science and Engineering Workshops*. 412 – 418. São Paulo, SP, Brazil.

Wada, H., J. Suzuki and K. Oba. 2007. A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture. In *Proceeding of IEEE International Conference on Services Computing*, ed. Liang-Jie (LJ) Zhang, Wil van der Aalst and Patrick C. K. Hung. 187 – 195.Salt Lake City, Utah, USA.

Wada, H., J. Suzuki and K. Oba. 2006. Modeling Non-Functional Aspects in Service Oriented Architecture. In *Proceeding of IEEE International Conference on Services Computing*. 222 – 229. Chicago, USA.

Wu, Z. and J. Luo. 2007. QoS-Resource Graph Model for Web Service Composition in Service Oriented Computing. In *Proceeding of Sixth International Conference on Grid and Cooperative Computing*. 411 – 416. Urumchi, Xinjiang, China.

Xie, H., A. Boukerche, M. Zhang and B.P Zeigler. 2008. Design of A QoS-Aware Service Composition and Management System in Peer-to-Peer Network Aided by DEVS. In *Proceeding of 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*. 285 – 291. Vancouver, BC, Canada.

Zeigler, B. P., T.G. Kim, and H. Praehofer. 2000. *Theory of Modeling and Simulation*. 2nd ed., New York, NY, USA: Academic Press

AUTHOR BIOGRAPHIES

HENGHENG XIE is a Master student in the Engineering Department at the University of Ottawa. He is a research assistant in the Paradise Research Lab. His research interests include distributed and parallel system, service composition and load balancing. His email is <hxie072@uottawa.ca>.

AZZEDINE BOUKERCHE is a Full Professor and holds a Canada Research Chair position at the University of Ottawa. He is the Founding Director of PARADISE Research Laboratory at University of Ottawa. Prior to this, he held a Faculty position at the University of North Texas, USA, and he was working as a Senior Scientist at the Simulation Sciences Division, Metron Corporation located in San Diego. He was also employed as a Faculty at the School of Computer Science McGill University, and taught at Polytechnic of Montreal. He spent a year at the JPL/NASA-California Institute of Technology where he contri-

buted to a project on the specification and verification of the software used to control interplanetary spacecraft operated by JPL/NASA Laboratory. His current research interests include wireless ad hoc and sensor networks, wireless networks, mobile and pervasive computing, wireless multimedia, QoS service provisioning, performance evaluation and modeling of large-scale distributed systems, distributed computing, large-scale distributed interactive simulation, and parallel discrete event simulation. His email is `<boukerch@site.uottawa.ca>.`

MING ZHANG is currently a Post-Doc Researcher at PARADISE Research Laboratory at the University of Ottawa. He obtained his Ph.D. at Arizona Center of Integrated Modeling and Simulation at University of Arizona. His research interests are modeling and simulation, large-scale distributed system and DEVS. His email is `<mizhang@site.uottawa.ca>.`