Available online at www.sciencedirect.com
ScienceDirect
Future Generation Computer Systems 24 (2008) 658–671
FGCS
FUTURE GENERATION COMPUTER SYSTEMS
www.elsevier.com/locate/fgcs

# Large scale agent-based simulation on the grid[☆]

Dan Chen[a], Georgios K. Theodoropoulos[a,*], Stephen J. Turner[b], Wentong Cai[b], Robert Minson[a], Yi Zhang[a,1]

[a] *School of Computer Science, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK*
[b] *School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore*

Received 7 February 2007; received in revised form 15 January 2008; accepted 21 January 2008
Available online 13 February 2008

**Abstract**

The development of many complex simulation applications requires collaborative effort from researchers with different domain knowledge and expertise, possibly at different locations. These simulation systems often require huge computing resources and data sets, which can be geographically distributed. In order to support collaborative model development and to cater for the increasing complexity of such systems, it is necessary to harness distributed resources over the Internet. The emergence of Grid technologies provides exciting new opportunities for large-scale distributed simulation, enabling collaboration and the use of distributed computing resources, while also facilitating access to geographically distributed data sets. This paper presents *HLA_Grid_RePast*, a middleware platform for executing large scale collaborating *RePast* agent-based models on the Grid. The paper also provides performance results from a deployment of the system between UK and Singapore.
© 2008 Elsevier B.V. All rights reserved.

*Keywords:* Grid; Agent-based systems; High Level Architecture; Distributed simulation

## 1. Introduction

Modelling and simulation is an essential tool in many areas of science and engineering, for example, for predicting the behaviour of new systems being designed or for analyzing natural phenomena. Simulation of complex systems has long been placed in the highly computation intensive world with computational requirements which far exceed the capabilities of a single computer. The term "distributed simulation" historically refers to the execution of discrete event simulation models on parallel and distributed platforms.

The last decade has witnessed an explosion of interest in distributed simulation techniques, not only for speeding up simulations but also as a strategic technology for linking simulation components of various types (e.g. discrete or continuous, numerical or discrete event etc.) at multiple locations to create a common virtual environment (e.g. battlefields, virtual factories and supply chains, agent-based systems, games etc). The culmination of this activity, which originated in military applications where battle scenarios were formed by connecting geographically distributed simulators via protocols such as DIS [8], has been the development of the High Level Architecture (HLA), a framework for simulator interoperability [7]. The HLA for Modelling and Simulation was developed as an IEEE standard to facilitate interoperability among simulations and promote reuse of simulation models. Using HLA, and the associated executable middleware, Runtime Infrastructure (RTI), a "large-scale distributed simulation" can be constructed by linking together a number of geographically distributed simulation components (or federates) into an overall simulation (or federation).

Such simulation systems often require huge computing resources and the data sets required by the simulation may

[☆] This is an extended version of the papers [G.K. Theodoropoulos, Y. Zhang, D. Chen, R. Minson, S.J. Turner, W. Cai, X. Yong, B. Logan, Large-scale distributed simulation on the grid, in: The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, Singapore, May 2006; Y. Zhang, G.K. Theodoropoulos, R. Minson, S.J. Turner, W. Cai, X. Yong, B. Logan, Grid-aware large-scale distributed simulation of agent-based systems, in: European Simulator Interoperability Workshop (Euro-SIW) 2005, Toulouse, France, 05E-SIW-047, July 2005]. The second paper received an Outstanding Paper Award.

[*] Corresponding author.
*E-mail addresses:* d.chen@cs.bham.ac.uk, chendan@pmail.ntu.edu.sg (D. Chen), gkt@cs.bham.ac.uk (G.K. Theodoropoulos), assjturner@ntu.edu.sg (S.J. Turner), aswtcai@ntu.edu.sg (W. Cai), rzm@cs.bham.ac.uk (R. Minson), yxz@cs.bham.ac.uk (Y. Zhang).

[1] Now with SAP, Belfast.

also be geographically distributed. For example, in a supply chain simulation involving different companies, the most up-to-date data will be in the individual companies. Furthermore, the development of such complex simulation applications usually requires collaborative effort from researchers with different domain knowledge and expertise, possibly at different locations. Typical examples would include transportation simulations, which are extremely challenging in requiring robust, accurate models of transportation infrastructures and their users for large metropolitan areas over time-scales ranging from minutes to years [30] or complex agent-based simulations, developed using different toolkits and interacting in different environment types [21]. In order to support collaborative model development and to cater for the increasing complexity of such systems, it is necessary to harness distributed resources over the Internet.

The emergence of Grid technologies provides an unrivalled opportunity for large-scale distributed simulation. While HLA enables the construction of large-scale distributed simulations using existing and possibly distributed simulation components, Grid technologies enable collaboration and provide mechanisms for the management of distributed computing resources where the simulation is being executed, while also facilitating access to geographically distributed data sets.

In the last few years, there has been an increasing interest in taking advantage of Grid technologies to execute HLA simulations over the Internet. Contributing to this global effort, this paper presents *HLA_Grid_RePast*, a prototype platform for executing large scale agent-based distributed simulations on a Grid. The work was undertaken as part of the DS-Grid[2] project, a collaborative project between the University of Birmingham in the UK and the Nanyang Technological University (NTU) in Singapore.

The *HLA_Grid_RePast* integrates two different middleware systems. At the bottom end, lies *HLA_Grid*, which has been developed by NTU as a middleware to support HLA simulations on the Grid [39]; at the top sits *HLA_RePast*, developed in Birmingham to support the execution of multiple interacting instances of *RePast* agent-based models within the HLA [19]. The *RePast* system [6] is a Java-based toolkit for the development of lightweight agents and agent models. It was developed at the University of Chicago's Social Science Research Computing division and is derived from the Swarm simulation toolkit. It has become a popular and influential toolkit, providing the development platform for several large multi-agent simulation experiments, particularly in the field of social phenomena. *RePast* has been assessed by [29,35] as the most effective development platform currently available for large-scale simulations of social phenomena. *HLA_Grid_RePast* allows large-scale *RePast* simulation systems to benefit from the advantages of both HLA and Grid technologies.

The rest of the paper is organised as follows: Section 2 summarises related work. Section 3 introduces the HLA and provides a short summary of the two existing constituent systems, namely *HLA_Grid* and *HLA_RePast*. Section 4 presents the architecture of *HLA_Grid_RePast* while Section 5 outlines the steps required for the deployment and execution of *HLA_Grid_RePast* systems on the Grid. Section 6 presents a quantitative performance evaluation of the system. Section 7 lists the key research challenges to be addressed in order to achieve seamless efficient distributed simulations on Grids. Finally, Section 8 concludes the paper and provides some ideas for future work.

## 2. Related work

Recent years have witnessed an increasing interest in taking advantage of Grid technologies to execute distributed simulations over the Internet, as manifested for instance by dedicated workshops[3] on this topic. Traditionally, HLA-based distributed simulations are conducted using a vendor-specific RTI software and federates with different RTI versions cannot cooperate with each other. To run a distributed simulation, the required software and hardware resource arrangements and security settings must be made before the actual simulation execution. Because of this inflexibility, it is not easy to run HLA-based distributed simulations across administrative domains and most of the distributed simulation applications are conducted inside a LAN. However, there is large demand for conducting HLA-based distributed simulation across a WAN (Wide Area Network). For example, to run an application where a specific federate can only be executed at a fixed site due to a specific resource requirement or security issues, the only solution is to enable HLA-based distributed simulation across WANs. The ability to conduct distributed simulation on a WAN also intrinsically enables large-scale applications utilizing the numerous WAN resources. In this section we outline the most representative work in this research area and evaluate it in a qualitative manner. An influential initiative in this area is the Extensible Modelling & Simulation Framework[4] (XMSF). XMSF makes use of Web-based technologies, applied within an extensible framework, that enables a new generation of Modelling & Simulation (M&S) applications to emerge, develop and interoperate. One of its important initiatives is to develop a web-enabled RTI [22,23,28]. Within the XMSF framework, multiple federates reside as web services on a WAN and the Federation's FOM is mapped to an XML tagset, allowing interoperation with other distributed applications supported by web services. The federates communicate using the Simple Object Access Protocol (SOAP) and the Blocks Extensible Exchange Protocol (BEEP), and reusability[5] of legacy federate code is well supported in XMSF.

Based on the concepts of XMSF, Xu and Peng [38] developed a Service Oriented EXtensible Modelling and Simulation Supporting Environment Architecture (SO-XMSSEA).

[2] http://www.cs.bham.ac.uk/research/projects/dsgrid.

[3] For example, DSGrid 2004 and DSGrid 2006 (http://pdcc.ntu.edu.sg/dsgrid06).

[4] http://www.movesinstitute.org/xmsf/xmsf.html.

[5] Reusability in this discussion refers to whether a normal federate's code can be reused upon the simulation frameworks.

Their SO-XMSSEA encapsulates HLA federates, federations and simulation supporting software as "simulation services" and presents them on the Internet through the support of GT4. Similarly, Chai et al. and Li et al. [4,18] have also presented a Service Oriented Infrastructure, namely Simulation Grid, to integrate Grid technology and HLA. The Simulation Grid provides HLA applications with the functionalities of dynamic resource sharing, fault tolerance and security. Work on this direction is also available in [40]. However, there is no performance analysis on any of these systems.

Wytzisk et al. proposed in [37] a solution that brings HLA and the Open GIS Consortium (OGC) standard together. The system provides external initialisation of federations, controlled start up and termination of federates, interactions with running federates, and access of simulation results by external processes.

Rycerz et al. designed a framework for HLA-based Interactive Simulations on the Grid [31]. The framework includes discovery and information indexing services and a single HLA Speaking service to manage multiple federates. Their work focuses on using Grid services to manage simulation federates, basically the migration of federates.

Another related work is IDSim, a distributed simulation framework based upon Open Grid Services Infrastructure [36], presented by Fitzgibbons et al. [10]. IDSim exploits Globus's Grid service data elements as simulation states to allow both pull and push modes of access. It was designed to ease the integration and deployment of tasks through inheritance. Finally, Iskra et al. [14,15] have looked at the execution of conventional, non-HLA distributed simulations (Time Warp kernel) on the Grid. Their system supports reusability of legacy simulation code.

The existing Grid- or Web-Enabled simulation frameworks have a variety of research focuses and follow different design approaches. XMSF was proposed to enhance the interoperability of HLA-based simulations. SO-XMSSEA aims at enabling the integration of HLA-based simulation and Grid technologies. Wytzisk et al.'s system eases the use of HLA-based simulations conforming to the OGC standard. IDSim uses OGSI to support extensibility of simulation systems. Rycerz et al.'s work mainly concerns facilitating migration of federates via Grid services, while Iskra et al.'s system adapts an optimistic Time Warp kernel to the Grid. Significantly different from the existing work, the *HLA_Grid_RePast* framework described in this paper focuses on supporting agent-based complex simulation and provision of HLA/RTI services via the Grid. The framework adopts a layered design, which allows the framework to be extended very conveniently, for example, to support fault tolerance. The framework can provide acceptable execution efficiency for complex systems, such as agent-based systems, and further performance optimisation can be envisioned. The middleware design supports reusability and user transparency to legacy federate code.

## 3. The constituent middleware layers

*HLA_Grid_RePast* glues together two different systems, *HLA_Grid* and *HLA_RePast* to enable the distributed execution of a *RePast* federation on the Grid. After a short introduction to the basic concepts of HLA and RTI, this section provides a summary of these two constituent systems of *HLA_Grid_RePast*. For a more detailed description of the two systems the reader is referred to [39,19] respectively.

### 3.1. High Level Architecture and Runtime Infrastructure

The HLA defines a software architecture for modelling and simulation. The HLA is designed to provide reuse and interoperability of simulation components. The simulation components are referred to as federates. A simulation federation can be created to achieve some specific objective by combining simulation federates. The HLA supports component-based simulation development in this way [7]. The HLA federation is a collection of federates (observers, live participants, simulators etc.) interacting with each other for a common purpose, for example wargaming. These federates interact with each other with the support of the Runtime Infrastructure (RTI) and the use of a common Federation Object Model (FOM). In the formal definition, the HLA standard comprises four main components: HLA Rules, Object Model Template (OMT), Interface Specification [12] and Federation Development and Execution Process (FEDEP) [13]. The HLA rules define the principles of HLA in terms of responsibilities that federates and federations must uphold. Each federation has a FOM, which is a common object model for the data exchanged between federates in a federation. The OMT defines the metamodel for all FOMs [16]. The HLA Interface Specification identifies the RTI services available to each federate and the functions each federate must provide to the federation. The FEDEP mainly defines a process framework for federation developers including the necessary activities to build HLA federations.

The HLA is an architecture defining the rules and interface, whereas the Runtime Infrastructure (RTI) is the software conforming to the HLA standard, that is used to support a federation execution. Fig. 1 gives an overview of an HLA federation and the RTI [16]. The RTI provides a set of services to the federates for data interchange and synchronization in a coordinated fashion. The RTI services are provided to each federate through its Local RTI Component (LRC) [9]. The RTI can be viewed as a distributed operating system providing services to support interoperable simulations executing in distributed computing environments [11]. A total of six service categories are defined in the specification, namely Federation Management, Declaration Management, Object Management, Ownership Management, Data Distribution Management and Time Management [9].

The RTI services are available as a library (C++ or Java) to the federate developers. Within the RTI library, the class *RTIAmbassador* [9] bundles the services provided by the RTI. A federate may invoke operations on the interface to request a service (federate-initiated service) from the RTI. The *FederateAmbassador* [9] is a pure virtual class that identifies the "callback" functions each federate is obliged to provide for (RTI-initiated) services. The federate developers need to

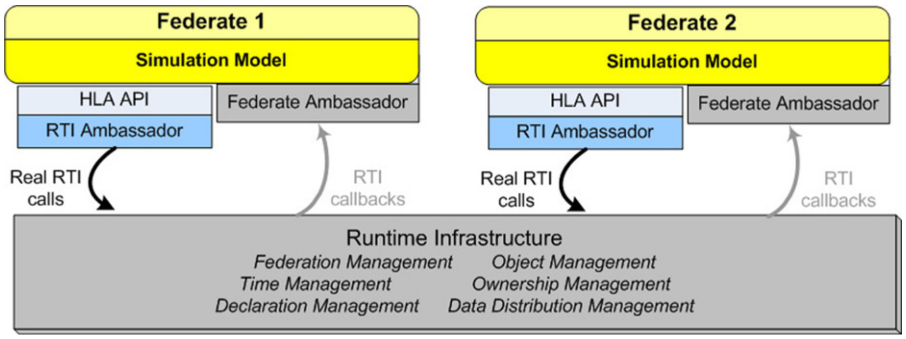


Fig. 1. Architecture of HLA-based distributed simulation.

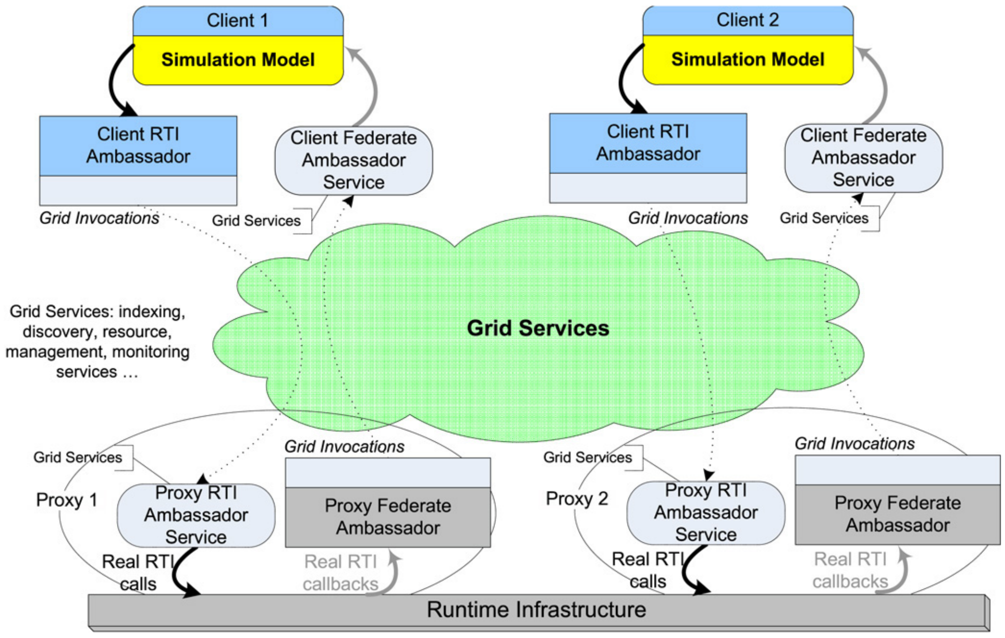

Client 1
Simulation Model
Client RTI Ambassador
Client Federate Ambassador Service
Client 2
Simulation Model
Client RTI Ambassador
Client Federate Ambassador Service
Grid Invocations
Grid Services
Grid Services: indexing, discovery, resource, management, monitoring services …
Grid Services
Proxy 1
Proxy RTI Ambassador Service
Proxy Federate Ambassador
Proxy 2
Proxy RTI Ambassador Service
Proxy Federate Ambassador
Real RTI calls
Real RTI callbacks
Runtime Infrastructure

Fig. 2. Architecture of *HLA_Grid*-based distributed simulation.

implement the FederateAmbassador. The callback functions provide a mechanism for the RTI to invoke operations and communicate back to the federate.

## 3.2. The HLA_Grid system

*HLA_Grid* is an infrastructure designed to extend the HLA to the Grid. In particular, it focuses on improving the interoperability and composability of HLA-compliant simulation components. The infrastructure is illustrated in Fig. 2.

*HLA_Grid* achieves interoperability between simulation federates using a Federate-Proxy-RTI architecture evolved from the Decoupled Federated Architecture proposed in [5]. In the context of *HLA_Grid*, an individual federate consists of a simulation model (namely the client from the user's perspective) and a *proxy* which acts on behalf of the client's federate code to communicate with the proxies of other clients through the underlying real RTI. Proxies are executed over remote Grid resources. A Grid-enabled HLA library provides the standard HLA API to the simulation model while translating the interactions into Grid service invocations.

*HLA_Grid* includes additional Grid services to support functions such as the creation of the RTI and the discovery of federations. The infrastructure hides the heterogeneity of the simulators, execution platforms, and how the simulators communicate with the RTI. The current *HLA_Grid* implementation has been developed in Java using the Globus Toolkit version 3 (GT3). The infrastructure conforms to the standard HLA interface to support Federation, Time, Object, Declaration and Ownership Management.

## 3.3. The HLA_RePast system

*HLA_RePast* is a middleware layer which enables the execution of a federation of multiple interacting instances of *RePast* models within the HLA as depicted in Fig. 3 [20]. *RePast* provides an inter-dependent collection of tools and structures, which are generally useful for the simulation of agents, and a sequential discrete event simulation kernel for the execution of the model. The main task of *HLA_RePast* is to detect the occurrence of events in the *RePast* model and communicate them to the underlying RTI in a consistent, reliable and transparent manner. To achieve this, *HLA_RePast*

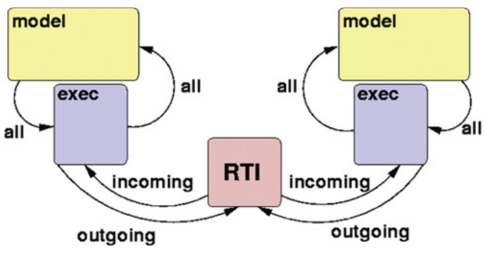


Fig. 3. An *HLA_RePast* federation.

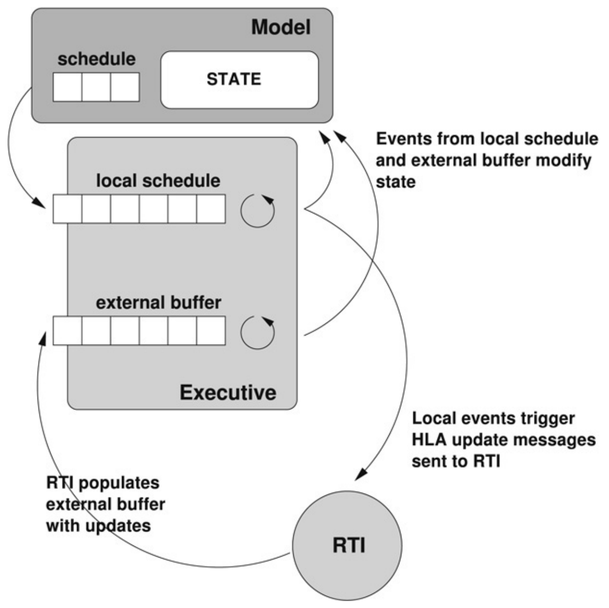

Model
schedule
STATE
local schedule
external buffer
Executive
Events from local schedule and external buffer modify state
Local events trigger HLA update messages sent to RTI
RTI populates external buffer with updates
RTI

Fig. 4. The *HLA_RePast* model–executive interface (a single *HLA_RePast* federate).

provides mechanisms for mapping *RePast* state-transitions to RTI events (via the *PublicObject* scheme described in Section 4.1), for conflict resolution and for integrating the *RePast* scheduling system with the RTI (as depicted in Fig. 4).

To ensure a maximum of transparency in scheduling, the normal *RePast* algorithm for stepping through the scheduler was replaced with an algorithm that constrains local advance in synchronization with the rest of the federation. The system does not execute a local event (a “BasicAction” in the local schedule) scheduled for time $T$ until it is possible to ensure that all remote events incoming from the RTI with timestamp $< T$ have been received. The RTI will then deliver a set of remote events to the federate, all with timestamps $\leq T$, these events are stored in the external event buffer. Once advance to $T$ is granted the buffer is flushed for all events with timestamp $< T$, resulting in a number of state modifications to the model. After this point the set of local events with timestamp $T$ can be executed via HLA update messages to the RTI, and the procedure repeats.

*HLA_RePast* is a promising infrastructure for developing complex agent-based models, and has been successfully used as a basis for a distributed and extensible architecture for modelling virtual crowds [21] and for the simulation of bacterial populations [17].

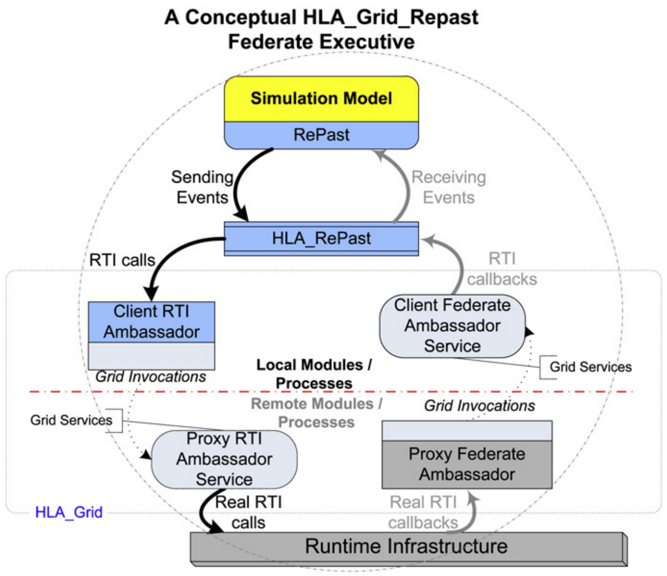


Fig. 5. Structure of *HLA_Grid_RePast*.

## 4. The *HLA_Grid_RePast* middleware system

A conceptual view of *HLA_Grid_RePast* is provided in Fig. 5. *HLA_Grid_RePast* uses the DMSO RTI NG 1.3 [9] as the underlying RTI software. Each federate executive consists of two major parts, one on the client side and another on the proxy side, which usually run on different machines. The client side includes the following modules: the *RePast* agent-based simulation model, the *HLA_RePast* middleware, the Client RTI Ambassador and the Client Federate Ambassador Service from *HLA_Grid*. These components usually run on a local machine (from the simulation model’s point of view).

On the proxy side, there are the Proxy RTI Ambassador Service and the Proxy Federate Ambassador which interact with the real RTI hosted by a remote machine. Both the Proxy RTI Ambassador Service and the Client Federate Ambassador Service are implemented as Grid services on different sides, shown as the round rectangle boxes in Fig. 5. The modules on both sides are coupled together to form a single compound federate executive.

Before the simulation starts, the Proxy RTI Ambassador Services should have been started, and each federate is associated with a Proxy RTI Ambassador Service. The identity of each federate’s Proxy RTI Ambassador Service will be passed to the corresponding Client RTI Ambassador. When the simulation starts, the Client Federate Ambassador Service of each federate will be initialised together with the corresponding Proxy RTI Ambassador Service.

During the simulation execution, the underlying *HLA_RePast* middleware translates the state changes (events) that occur in the simulation model into the corresponding RTI service calls. Then the RTI calls are passed to the Client RTI Ambassador, and the latter convert these RTI calls once again into Grid service invocations to access the remote Proxy RTI Ambassador Service at the proxy side.

Finally, the Proxy RTI Ambassador Service will interact with the real RTI by executing the real RTI calls with respect to the client side. The return values of a RTI call will be sent back in the form of the Grid service call's return value, while the runtime exceptions will be sent back by means of Apache Axis faults.

On the other hand, callbacks [9] from the RTI to a federate are translated into invocations of the Client Federate Ambassador Service by the Proxy Federate Ambassador. The Client Federate Ambassador Service is responsible for conveying these callbacks to the *HLA_RePast* middleware, and the latter should then convert the federate calls into *RePast* events and put them into the local event scheduler (see Fig. 4).

A number of issues had to be addressed to integrate *HLA_Grid* and *HLA_RePast* into *HLA_Grid_RePast*. Indicatively, the sections below describe two examples, Object Encoding and Remote Exception Handling. The former is required by the SOAP protocol while the latter was developed for performance improvement.

### 4.1. Object encoding

The object encoding scheme of *HLA_RePast* has been modified to cooperate with *HLA_Grid*. *HLA_RePast* uses a *PublicObject* scheme to translate Java expressions into HLA function calls [19]. Java objects in the *PublicObject* scheme may be transferred between federates during simulation execution. In *HLA_RePast*, the Java objects are encoded into Java Byte arrays. However, Java Byte arrays cannot be used directly within *HLA_Grid*, which, like other Grid services, uses SOAP as the communication protocol between clients and servers. SOAP is based on XML which does not support binary data. The problem can be solved by modifying *HLA_Grid* to support either the *SOAP With Attachments* (SwA) proposal [1] or the *WS-Attachments* proposal [24]. Such an approach would require modifications to the GT3 and would affect the interoperability of *HLA_Grid_RePast*. Instead of modifying GT3, *HLA_Grid_RePast* employs an alternative object encoding scheme for Java objects as follows:

- First, the Java object is encoded into a Java byte array as in *HLA_RePast*.
- Then, the base64 encoding scheme (from the Apache project) is used to encode the Java byte array into a Java String Object.
- Finally, a Java String is generated from the Java Byte array and passed to *HLA_Grid* for encapsulation in a SOAP message.

### 4.2. Exception handling

In *HLA_Grid_RePast*, Java exceptions generated in Grid service calls are packed into Apache Axis faults and passed back to the client side. *HLA_Grid_RePast* adopts a design of remote exception handling. In the current design, Java exceptions are often treated as return values of RTI calls, and *HLA_RePast* makes heavy use of RTI calls. In many cases, the exceptions are discarded immediately without further handling. This behaviour causes very small overhead when the simulation federates and the RTI Ambassadors execute on the same machine. However, this is not the case in *HLA_Grid*, where the federate simulations and the RTI Ambassadors located in different machines are connected via the Grid. In this case, dealing with exceptions raised by RTI calls wastes substantial network bandwidth. To tackle this drawback, in *HLA_Grid* such exceptions are registered with the Proxy RTI Ambassador Services after establishing the connection between the simulation model and Proxy RTI Ambassador Service. When RTI calls generate exceptions, the Proxy RTI Ambassador Service will only pass unregistered exceptions back to the client side. In this way, registered RTI call exceptions are handled remotely at the proxy side. Section 6 demonstrates the performance improvement achieved by this design.

## 5. Deployment and execution of *HLA_Grid_RePast* systems

Within the current implementation of the *HLA_Grid_RePast* system, two Grid service "factories" [32] need to be deployed prior to executing a distributed agent-based simulation on the Grid; these are *ClientFactoryService* (CFS) and *ProxyFactoryService* (PFS). The two factories should be started by the system administrator and persist throughout the simulation. The *ClientFactoryService* is located at the client side (see Fig. 5) and supports the *RePast* simulation model and the instantiation of the Client Federate Ambassador Service. Similarly, the *ProxyFactoryService* resides in the proxy side and is responsible for creating the Proxy RTI Ambassador Service. The *ClientFactoryService* can only be accessed by the authorised simulation users verified through GT3's built-in Certification Authorisation [32], while the *ProxyFactoryService* is user transparent and merely responds to the authorised clients to eventually generate services for the *RePast* models.

Fig. 6 illustrates the procedure of initiating an *HLA_Grid_RePast* system. First, the simulation user simply needs to request the preselected *ClientFactoryService*[6] and pass the information of the target *RePast* model. Once this step is successfully accomplished, the subsequent steps will continue automatically and transparently to the user. For the second step, the CFS instantiates an instance of Client Federate Ambassador Service and invokes the designated *RePast* model. The *RePast* model then requests the pre-specified remote *ProxyFactoryService*.[7] After granting the *RePast* model's request, the PFS instantiates an instance of Proxy RTI Ambassador Service and starts the corresponding proxy process. Finally, the Proxy RTI Ambassador Service will be provided to the *RePast* model and the Client Federate Ambassador Service to the remote proxy. From here onwards, these processes and services collaborate with each other to execute an *HLA_Grid_RePast* federate executive to allow this *RePast* model interoperate with its peer *RePast* models in the distributed simulation.

[6] For example, http://pdcc.ntu.edu.sg/ogsa/services/hlagrid/client/ClientFactoryService.

[7] For example, http://escience.bham.ac.uk/ogsa/services/hlagrid/proxy/ProxyFactoryService.

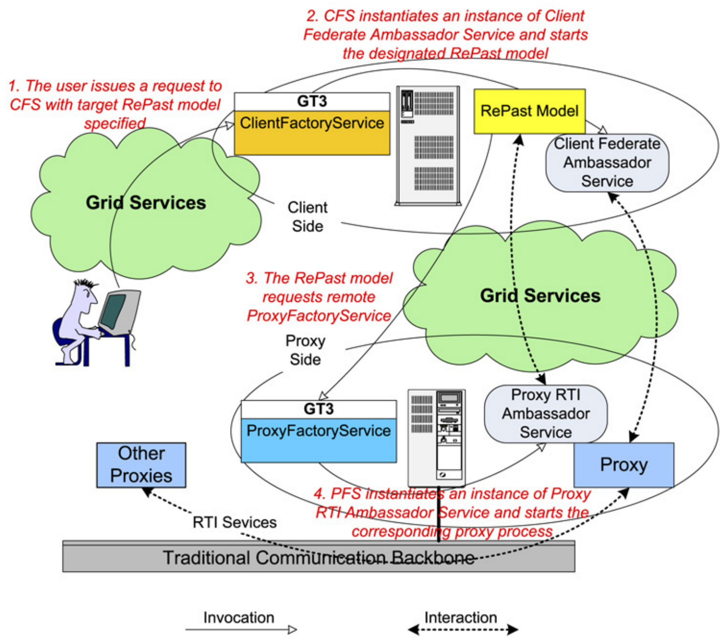

2. CFS instantiates an instance of Client Federate Ambassador Service and starts the designated RePast model
1. The user issues a request to CFS with target RePast model specified
GT3
ClientFactoryService
RePast Model
Client Federate Ambassador Service
Grid Services
Client Side
3. The RePast model requests remote ProxyFactoryService
Grid Services
Proxy Side
GT3
ProxyFactoryService
Proxy RTI Ambassador Service
Proxy
Other Proxies
4. PFS instantiates an instance of Proxy RTI Ambassador Service and starts the corresponding proxy process
RTI Sevices
Traditional Communication Backbone
Invocation
Interaction

Fig. 6. Execution of *HLA_Grid_RePast* based simulation.

The *HLA_Grid_RePast* has the additional advantage over *HLA_RePast* of dealing with network security issues. To execute two federates and an RTIEXEC process (similar to the scenario in Fig. 12) using *HLA_RePast* together with DMSO RTI NG 1.3, there should be one machine, for example, in Singapore and another two, for example, in Birmingham. The machine in Singapore has to communicate with both remote machines in Birmingham, something that needs the setting of two new firewall rules in Singapore. If more federates are added in Birmingham, the more firewall rules should be applied. In contrast, when using *HLA_Grid_RePast*, the machine in Singapore only needs to communicate with the machine running the *HLA_Grid* proxy service in Birmingham, therefore only one firewall rule is needed even if more federates are involved.

## 6. Experiments and evaluation

To evaluate the robustness and the performance of the system, we have conducted a number of experiments[8] in both a LAN environment and a WAN environment between the UK (Birmingham) and Singapore. The DMSO RTI NG 1.3 with Java bindings is used in the experiments. The LAN environment is a Grid-enabled PC cluster at Birmingham consisting of one master node and 54 worker nodes.[9] Each node has 2 Gbytes of memory and two Intel Xeon 3 GHz processors while connectivity is provided by a Gigabit Ethernet Switch. The WAN environment is constructed by the Birmingham cluster and a server at Singapore over Internet. The Singapore server has 1 Gbytes of memory and four Intel Pentium III (Katmai) processors.

### *6.1. Communication overhead*

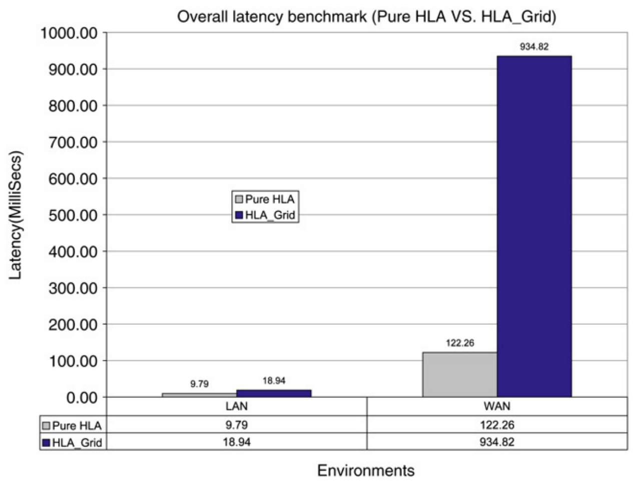
In order to investigate the communication overhead incurred by the underlying *HLA_Grid* infrastructures and establish a base line for future experiments, the Java-based latency benchmark from DMSO's RTI package has been built and tested under LAN and WAN environments. The benchmark measures the performance of the system in terms of the latency of federate communications, as follows: One federate sends an attribute update, and upon receiving this update, the other federate sends it back to the sending federate. The elapsed time of this communication is calculated using the real-time taken at the sending and reflecting federates. The two pure HLA federates run on two separate nodes, and the two *HLA_Grid* federates (see Fig. 2) also run on two separate nodes while they require two additional proxies running on two additional nodes to service them.

The results obtained are shown in Fig. 7. The latency benchmark over LAN shows that *HLA_Grid* incurs about twice the overhead of the pure HLA in a cluster. This is mainly due to the use of the GT3 communication backbone, the encoding/decoding of parameters/result etc. In the case of the WAN environment, such communication using GT3 becomes very costly. Comparing the latencies of *HLA_Grid* over WAN and LAN, the latency of *HLA_Grid* over WAN is ∼50 times more than the latency of *HLA_Grid* over LAN. It can also be observed that the latency of *HLA_Grid* (∼935 ms) is ∼7 times more than that of the pure HLA case under the WAN environment.

### *6.2. Performance of HLA_Grid_RePast in a LAN environment*

As a case study for the evaluation of *HLA_Grid_RePast*, we have developed a *RePast* federation using TILEWORLD [27] as

[8] The experimental results presented in this section are averaged from the outputs of a number of runs, and the deviations are very insignificant.
[9] http://www.ep.ph.bham.ac.uk/cluster.



| | LAN | WAN |
|---|---|---|
| Pure HLA | 9.79 | 122.26 |
| HLA_Grid | 18.94 | 934.82 |

Fig. 7. Latency benchmark over LAN and WAN.

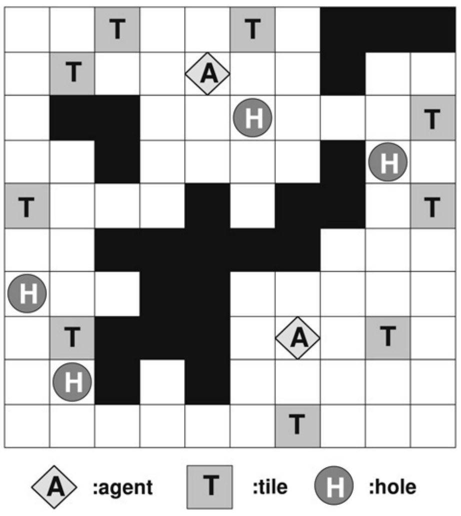
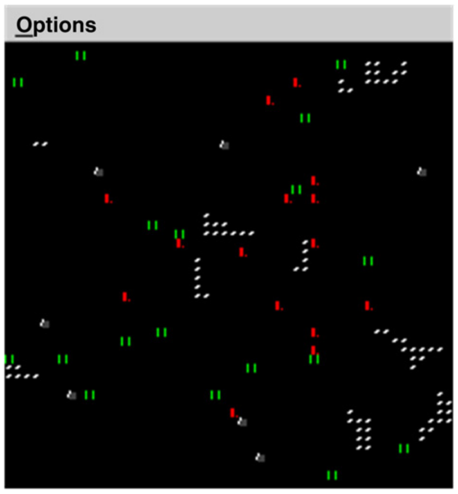


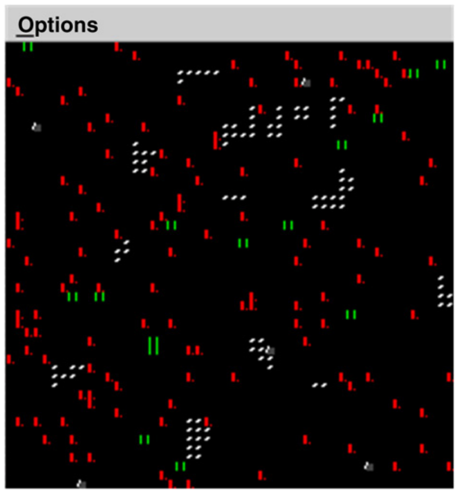
Fig. 8. A 10 × 10 TILEWORLD.

a test case. TILEWORLD is a well-established testbed for agent-based systems. It includes a grid-like environment consisting of tiles, holes and obstacles, and agents whose goal is to score as many points as possible by pushing tiles to fill in the holes (Fig. 8). Whenever a hole is filled, the agent who dropped the last tile will be given a score as the initial depth of the hole. The environment is dynamic: tiles, holes and obstacles appear and disappear at rates controlled by the simulation developer. TILEWORLD has been used to study commitment strategies (i.e., when an agent should abandon its current goal and replan) and in comparisons of reactive and deliberative agent architectures [27].

For *HLA_Grid_RePast*, a TILEWORLD simulation originally developed for *HLA_RePast* has been used, as described in [19]. In general, an agent-based federation can be constructed by one *environment federate* containing the tiles, holes and obstacles of the model and one or more *agent federates*. The complexity of the simulated system increases considerably with the number of agents.



(a) A Tileworld scenario with 16 agents.



(b) A Tileworld scenario with 128 agents.

Fig. 9. Snapshot of two TILEWORLD scenarios.

### 6.2.1. *Performance evaluation in a LAN environment using a single agent federate*

This section presents the experiments in a LAN environment with one agent federate simulating different numbers of agents. The performance of *HLA_Grid_RePast* and *HLA_RePast* is compared to evaluate the overhead introduced by the different layers of the middleware (*HLA_Grid* and *HLA_Grid_RePast*). In this case, the entire federation is constructed by an environment federate and a single agent federate hosting all agents. The number of agents varies from 1 to 128. Fig. 9 presents the snapshot of two examples of TILEWORLD scenarios.

For *HLA_RePast*, the two *RePast* federates run on two separate nodes. For *HLA_Grid_RePast*, two more nodes are required to host the respective *HLA_Grid* proxies which serve the two *RePast* models. In all experiments, the RTIEXEC is running on a separate node. The experimental results obtained are reported in Fig. 10. For *HLA_Grid_RePast*, two sets of experiments have been performed, one with local (*HLA_Grid_RePast(L)*) and another with remote (*HLA_Grid_RePast(R)*) exception handling.

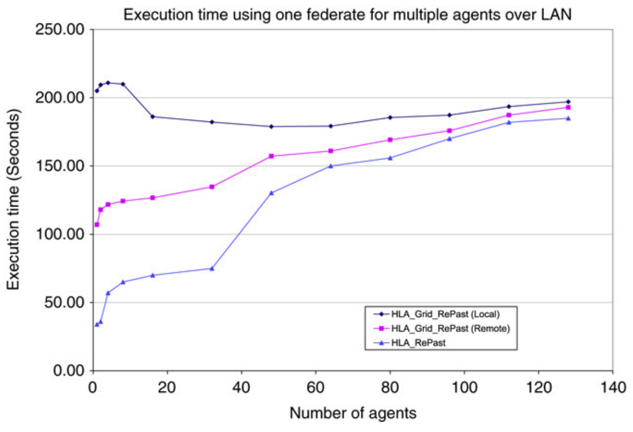


Fig. 10. Execution times of *HLA_Grid_RePast*- and *HLA_RePast*-based simulations using a single Agent Federate.

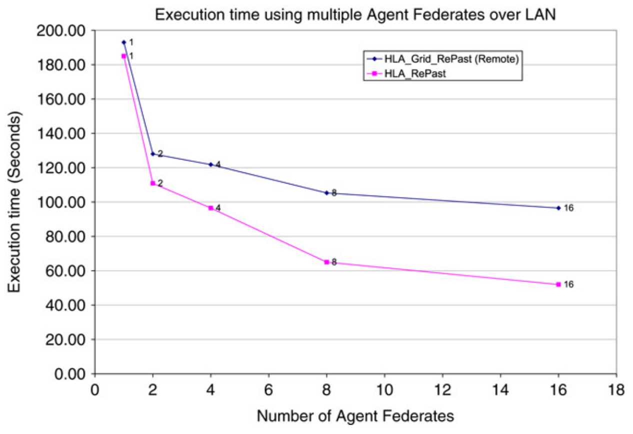


Fig. 11. Execution times of *HLA_Grid_RePast*- and *HLA_RePast*-based simulations with multiple agent federates (128 agents).

Clearly, the results show that the performance of *HLA_Grid_RePast* with local exception handling is lower compared with that with remote exception handling. It should be noted that when the number of agents is small (less than 32 in our experiments), both implementations of *HLA_Grid_RePast* exhibit a significantly higher overhead compared to *HLA_RePast*. This is attributed to *HLA_RePast*'s heavy use of RTI functions that may generate Java exceptions which in turn are just caught and discarded. Thus, the system spends most of its time sending and receiving RTI exceptions, which have no effect on the simulation. As communication and computation tasks in *HLA_Grid_RePast* are carried out concurrently by separate Java threads, the communication time and computation time overlap. In the case of *HLA_Grid_RePast(L)*, the number of exceptions to be handled is significantly high, causing an irregular pattern in the execution times until the number of agents becomes large enough. As the number of agents increases further, the computation load rather than communication load becomes the determining factor, and the total execution time for *HLA_Grid_RePast* approximates that of *HLA_RePast*.

From the experimental results presented in Fig. 10, we can perform the $t$-Test (paired, two-tailed distribution) on the execution times (number of agents greater than 60) between *HLA_Grid_RePast(L)* and *HLA_RePast* ($t_1$) as well as between *HLA_Grid_RePast(L)* and *HLA_Grid_RePast(R)* ($t_2$). The outputs can be written as: $t_1 = 0.009127184$ and $t_2 = 0.003355111$, which indicate that the differences of execution times amongst these systems become insignificant when the number of agents is high enough.

### *6.2.2. Performance evaluation in a LAN environment using multiple agent federates*

This series of experiments focuses on speedup and aims to investigate and compare the performance of *HLA_RePast* and *HLA_Grid_RePast* systems (*HLA_Grid_RePast(R)* adopted) in a LAN environment by evenly distributing 128 agents in multiple federates (1–16). Encapsulating a large number of agents in a single federate often introduces a bottleneck. Distribution of agents aims to alleviate the system load of each Agent Federate. Obviously, as the number of agent federates increases, the computational load of each federate becomes lighter, resulting in higher overall communication amongst the distributed federates.

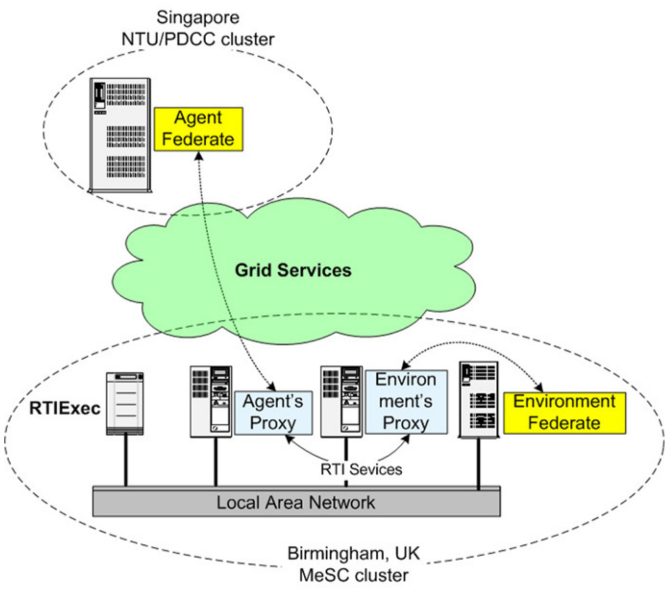


Fig. 12. TILEWORLD configuration for *HLA_Grid_RePast* in a WAN environment.

Fig. 11 gives the execution times of *HLA_RePast* and *HLA_Grid_RePast* with different numbers of agent federates. The execution times of both *HLA_Grid_RePast*- and *HLA_RePast*-based simulations decrease with the number of agent federates. However, as the computational load in each federate decreases, the communication overhead becomes more and more significant. Since *HLA_RePast* incurs less latency than *HLA_Grid_RePast*, it gains more speedup.

As shown in Fig. 11, *HLA_Grid_RePast* yields a speedup pattern similar to *HLA_RePast*, while incurring slightly more overhead. In *HLA_Grid_RePast*, each federate requires one node for the federate itself and one for the proxy service. Such extra complexity contributes to the additional communication cost.

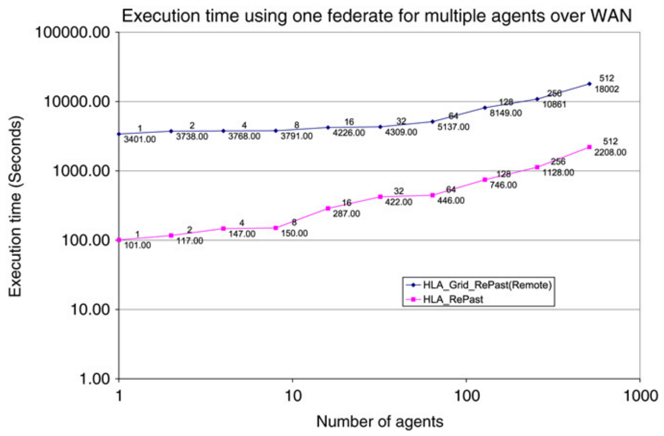


Fig. 13. Execution time of *HLA_Grid_RePast*- and *HLA_RePast*-based simulations with a single Agent Federate over WAN.

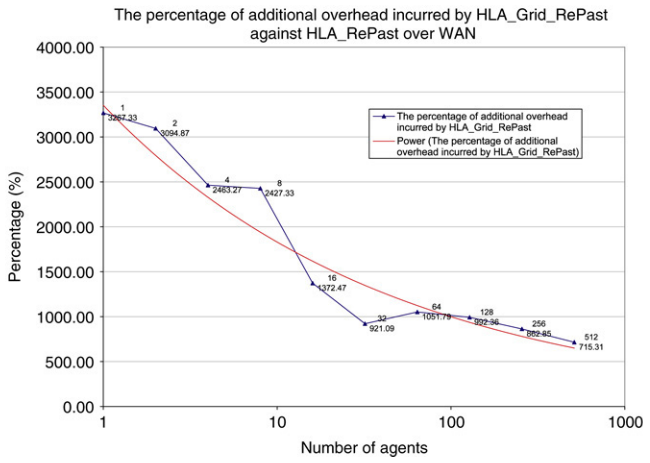


Fig. 14. The percentage of additional overhead incurred by *HLA_Grid_RePast* against *HLA_RePast* over WAN.

### 6.3. Performance in a WAN environment

Experiments for evaluating the performance of *HLA_RePast* and *HLA_Grid_RePast* have also been conducted in a WAN environment. In these experiments, the RTIEXEC and the environment federate are in Birmingham while the agent federate is in Singapore, as illustrated in Fig. 12. Two *HLA_Grid* proxy services are also hosted in Birmingham for the environment and agent federates respectively.

Similar to the first set of experiments in the LAN environment, only one agent federate is used with the number of agents ranging from 1 to 512. The results for a WAN environment are depicted in Fig. 13 (logarithmic scale applies in both $X$-axis and $Y$-axis).

Comparing Figs. 13 and 10, it can be observed that simulations take much longer to finish in the WAN environment. This is not surprising, as the bandwidth is much smaller and the communication latency much higher in the WAN environment. As shown in Fig. 13, the most noticeable difference is that *HLA_Grid_RePast* performs much worse than *HLA_RePast* in the WAN environment. Nevertheless, when the number of agents becomes larger, the total execution time with *HLA_Grid_RePast* tends to be closer to that with *HLA_RePast*.

Fig. 14 highlights the percentage of additional overhead of *HLA_Grid_RePast* against *HLA_RePast*. We write the execution times of *HLA_Grid_RePast* and *HLA_RePast* as $T_{hgr}[i]$ and $T_{hr}[i]$ respectively, and the percentage of additional overhead is calculated as $100 \times ((T_{hgr}[i] - T_{hr}[i]) / T_{hr}[i])$. With one agent, the total execution time with *HLA_Grid_RePast* is ~32.6 times more than that with *HLA_RePast*. This difference diminishes rapidly with the number of agents (or in other words with the complexity of the simulation) becoming ~7.2 with 512 agents. As the same computation loads (for an equal number of agents) apply in both environments, the difference in execution times between *HLA_Grid_RePast* and *HLA_RePast* can be attributed to the communication overhead, which is much higher in *HLA_Grid_RePast* (see Fig. 7).

### 6.4. Analysis and further optimisation

From the experiments presented in previous sections, it is apparent that in a WAN environment *HLA_Grid_RePast* incurs much higher overhead than *HLA_RePast*. As there is no difference between the *RePast* models used for the two systems, the extra overhead of the former is attributed to the communication load. In order to identify the cause of the performance gap, additional measurements have been taken from the experiments in terms of number/types of RTI calls, which determine a federate's communication pattern, and the underlying networking traffic. In the WAN environment, each RTI call from the agent federate has to be conveyed over Grid to the proxy to be executed. RTI calls can be classified into normal RTI Ambassador calls (normal calls) and RTI Ambassador ticks (ticks) [9].

In the above experiments, the RTI related execution of an agent federate has 100 iterations, and each involves the following:

1. *Callbacks:* Process callbacks delivered by the RTI.
2. *Normal* RTI *Ambassador calls:* Generate attribute updates of RTI object instances.
3. *Time Advancement:* Issue time advancement request to the RTI then keep issuing *ticks* until the request is granted.

Comparing the logs of *HLA_Grid_RePast* experiments under the LAN and WAN environments, we confirm that the agent federate issues a similar number of normal RTI calls in both environments (e.g., for 128 agents: 11060 (LAN) vs. 11745 (WAN)). In contrast, there exist a much larger number of ticks in the WAN (e.g., for 128 agents: 990 (LAN) vs. 4120 (WAN)), most of which result from waiting for the remote RTI to grant time advancement requests to the *RePast* model.

Another measurement collects data at the network layer. The network packets (TCP) generated between clients and proxies under WAN and LAN environments have been monitored. The Syn/Fin messages of the TCP protocol are of particular interest as those are handshake packets to establish TCP connections. In experiments with *HLA_RePast* on a WAN, as the number of agents increases, the number of Syn/Fin packets remains constant (100). The same TCP connections can be reused for a number of RTI calls. For experiments with *HLA_Grid_RePast* on a WAN, the corresponding statistics are dramatically different. Within the current *HLA_Grid* architecture implemented using GT3 [39], a new TCP connection over the Internet will be established for each RTI call. Consequently, *HLA_Grid_RePast* generates many more Syn/Fin Packets (e.g., for 128 agents on a WAN: 75600). Such costly operations have an overwhelming effect on the execution time of *HLA_Grid_RePast* over the Internet. For all experiments with *HLA_Grid_RePast*, it has been observed that over the Internet the time spent in establishing all TCP connections is more than half of the entire simulation time, while within a LAN it is negligible (e.g., for 128 agents: ∼2 seconds (LAN) vs. ∼6967 seconds (WAN)).

From the above, we can conclude that the performance degradation of *HLA_Grid_RePast* in the WAN environment is mainly due to the large number of messages related to RTI calls. Significant overhead has been introduced by establishing excessive TCP connections over the Internet.

Under the current GT3-based implementation, using a middleware layer may provide a solution to the problem by reducing the number of messages over the Grid including both the normal RTI calls and the ticks. For instance, in each iteration, we can aggregate the attribute updates of object instances into an "aggregated message" at the client side. After the aggregated message is transmitted over the Grid, the proxy can decompose it and execute multiple attribute updates accordingly. With regard to the ticks, immediately after a federate has requested time advancement, its proxy can keep ticking and wait for time advancement granted. In the original *HLA_Grid_RePast*, a proxy will issue an RTI tick only when the simulation model initiates a tick at the client side. In this way, the number of Grid messages can be significantly reduced. As the federates used in the previous experiments consists of one hundred iterations, ideally there will be only one hundred aggregated messages and one hundred tick messages to be delivered over the Grid. Iskra et al. have successfully employed a similar message aggregation approach to improve the performance of their Time Warp kernel on the Grid [15].

We repeated the experiments described in Section 6.3 using the *HLA_Grid_RePast* enabled with message aggregation (*HLA_Grid_RePast_Message_Aggregation*). Fig. 15 presents the execution times of *HLA_Grid_RePast_Message_Aggregation* against *HLA_RePast*, and Fig. 16 highlights the percentage of additional overhead of *HLA_Grid_RePast_Message_Aggregation*.

Comparing the results presented in Fig. 15 to those in Fig. 13, we can observe that message aggregation has dramatically reduced the communication overhead incurred by *HLA_Grid_RePast*. The total execution time of *HLA_Grid_RePast_Message_Aggregation* is now ∼16.7 times that of *HLA_RePast* with one agent, and the percentage of additional overhead diminishes rapidly with the number of agents. It becomes less than ∼3.2 times the execution time of *HLA_RePast* with 512 agents. These results indicate that the overhead of a higher payload in each aggregated message

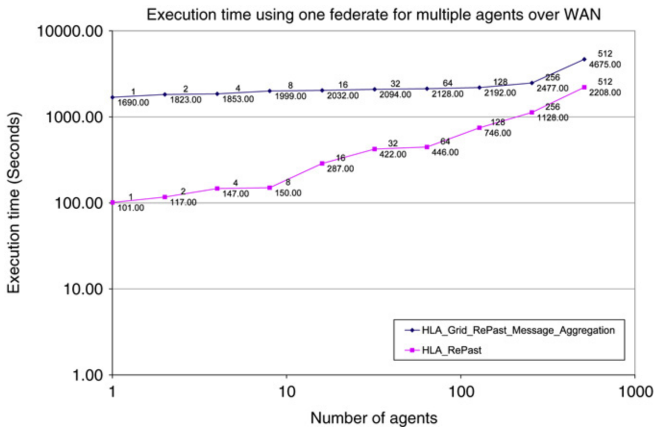


Fig. 15. Execution time of *HLA_Grid_RePast_Message_Aggregation*- and *HLA_RePast*-based simulations with a single Agent Federate over WAN.

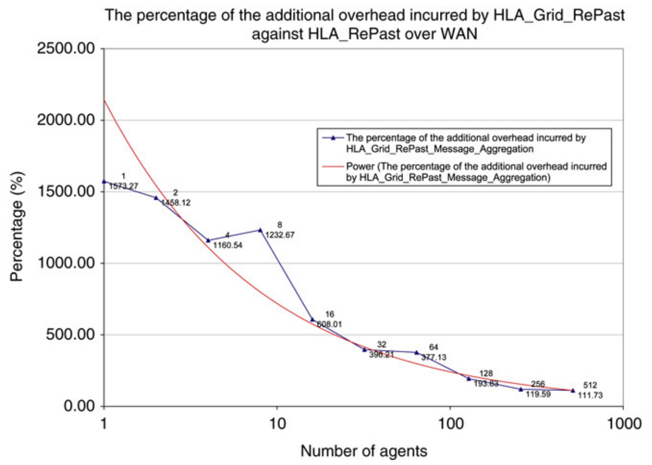


Fig. 16. The percentage of additional overhead incurred by *HLA_Grid_RePast_Message_Aggregation* against *HLA_RePast* over WAN.

can be compensated by reducing the number of TCP connections, therefore significant performance improvement can been achieved.

Clearly, with the current testbed used for the WAN experiments (see Fig. 12), the performance of *HLA_RePast* is expected to be superior to *HLA_Grid_RePast*. Both federates in the federations actually execute the same compute intensive simulation models, with the agent federate at the Singapore node and the environment federate at the Birmingham node. However, the cost incurred by the communication between Singapore and Birmingham using *HLA_Grid_RePast* federates is much higher than that using *HLA_RePast* federates. As the overall performance of the simulation systems are determined by the computation cost and the communication cost, the performance of *HLA_Grid_RePast* will tend towards that of *HLA_RePast* with the increasing complexity of the agent/environment federates. However, *HLA_RePast* does not have the benefits of using the Grid technologies as described in Section 1.

## 7. A Vision for the future

The Grid-aware HLA is a fast evolving technology which offers enormous potential for large scale distributed simulation on the Grid. *HLA_Grid_RePast* as a proof-of-concept has illustrated the difficulties but also the potential of such an approach for complex agent-based simulations. The grand vision is a "Grid plug-and-play distributed simulation system", a distributed collaborative simulation environment where researchers with different domain knowledge and expertise, possibly at different locations, develop, modify, assemble and execute distributed simulation components over the Grid. However a number of challenging problems have to be addressed before this vision is realised.

### 7.1. Automated model discovery and composability

Composability as defined by Petty et al. [26] is "the capability to select and assemble simulation components in various combinations into simulation systems to satisfy specific user requirements". While the FOM and RTI provide interoperability at the communication level, there is little support for interoperability at the semantic level; as a result, simulation development lead times are often on the order of months, since the interpretation of data by each federate must be fully understood and carefully checked for consistency by the simulation developer. Thus, there is a clear need for automated federate discovery and configuration in a Grid environment.

For automated federate discovery and configuration, appropriate ontologies and languages must be defined to represent the metadata of federates and orchestrate simulations by matching user requirements with appropriate federates. Each federate can manifest itself as a Grid service for use by the simulator. The ontologies should represent descriptions, classifications of descriptions, and constraints related to valid configurations of simulations and should support both searching for simulation models and semantic matching of component models with simulation goals. To achieve this, in addition to frameworks at the syntactic level (such as XMSF and BOM [2]) ontologies for the description of the internal behaviour of federates are required, as well as appropriate reasoning and matchmaking engines. Although there have been efforts to describe the inner mechanisms of composite or stateful services, so far these have been limited [33]. Not without reason this has been identified as one of the grand challenges in Modelling and Simulation [25].

### 7.2. Service oriented architectures for simulation

Most existing work on executing HLA-based distributed simulations on the Grid surveyed in Section 2, including *HLA_Grid_RePast* share a common feature: the Grid middleware is employed to perform the tasks of resource management (including scheduling, load balancing and monitoring), coordination of simulation execution, and security while the existing HLA/RTI system is used to perform simulation related tasks such as synchronization, time management and data distribution management.

Such an approach has a number of deficiencies, as the work with *HLA_Grid_RePast* has confirmed. The dependency on an existing HLA/RTI system, which was not developed for execution on the Grid, requires the adaptation of both the simulation applications and the HLA/RTI system to the Grid. This inevitably introduces additional overheads, restricts the simulation application development, and limits the availability of the simulation services provided by the HLA/RTI. In addition, the simple integration of the Grid middleware and the HLA/RTI cannot fully exploit benefits such as loose coupling, heterogeneity, and transportation protocol independence that would be provided by a Service Oriented Architecture (SOA). The XMSF framework has demonstrated the advantages of SOA for distributed simulations, however it is based on web rather than Grid technologies. Work has already commenced to convert the underlying infrastructure of *HLA_Grid_RePast* (*HLA_Grid*) to an SOA.

### 7.3. Resource management

Another important issue is the need for resource and load management mechanisms to support large-scale simulation in a Grid environment. A large-scale simulation executing in a Grid environment may need computing resources from many different organisations and the availability of these computing resources may change during the execution of the simulation. To meet the real-time requirements demanded by interactive simulations and the performance requirements demanded by analytical simulations, sophisticated resource management mechanisms are required which are missing in current HLA implementations. These mechanisms should go far beyond the simple migration of federates (e.g., [3]) and provide support for issues such as load monitoring, dynamic load balancing, check-pointing etc [34].

## 8. Conclusions and future work

The emergence of Grid technologies provides exciting new opportunities for large-scale distributed simulation. This paper has presented *HLA_Grid_RePast*, a middleware platform for the execution of collaborating *RePast* on the Grid. The system presents *RePast* models as services and facilitates the flexible construction and execution of large scale agent federations over the Grid. It has successfully supported reusability and user transparency, providing a relatively simple workflow for deployment and usage.

The experimental results show that communication costs are crucial for the performance of such simulations. However the additional overhead introduced in the system as a result of communication steadily becomes less significant as the complexity of the simulation models increases, suggesting that the deployment of the Grid can indeed offer performance gains for large scale computation bound simulations.

Future work will seek to further improve the performance of the system using alternative application containers, such as the Apache Tomcat, as well as to migrate *HLA_Grid_RePast* from GT3 to Globus Toolkit 4.1.1 (GT4.1.1). Further reduction of the communication overhead over WAN can be envisioned with the persistent TCP/IP connections supported by GT4.1.1.

## Acknowledgements

A number of people have helped in many different ways to make this work possible. In particular we would like to thank Krzysztof Kostienko, Ben Stone, Steve Pillinger, Jim Nichols, Irene Goh, Tan Joo Sing. This work was funded by the UK e-Science Programme (Grant No. GR/S82862/01).

## References

[1] J. Barton, S. Thatte, H.F. Nielsen, SOAP messages with attachments, W3C note, December, 2000.

[2] Base Object Model. http://www.boms.info/, January, 2007.

[3] W. Cai, Z. Yuan, M.Y.H. Low, S. Turner, Federate migration in HLA-based simulation, Future Generation Computer Systems 21 (1) (2005) 87–95.

[4] X. Chai, H. Yu, Z. Du, B. Hou, B. Li, Research and application on service oriented infrastructure for networkitized M&S, in: The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, Singapore, May, 2006.

[5] D. Chen, S.J. Turner, B.P. Gan, W. Cai, J. Wei, A decoupled federate architecture for distributed simulation cloning, in: 15th European Simulation Symposium, ESS 2003, Delft, Netherlands, October, 2003, pp. 131–140.

[6] N. Collier, RePast : An extensible framework for agent simulation. http://www.econ.iastate.edu/tesfatsi/RepastTutorial.Collier.pdf, January 2007.

[7] Judith S. Dahmann, Frederick Kuhl and Richard weatherly, standards for simulation: As simple as possible but not simpler: The high level architecture for simulation, Simulation: Transactions of The Society for Modeling and Simulation International 71 (6) (1998) 378–387.

[8] DIS Steering Committee. The DIS Vision, A map to the future of distributed simulation, Technical Report IST-SP-94-01, Institute for Simulation and Training, Orlando, Florida, USA, 1994.

[9] DMSO/DOD, RTI 1.3 Next generation programmer's guide version 5, Alexandria, VA, USA, 2002.

[10] J.B. Fitzgibbons, R. Fujimoto, D. Fellig, D. Kleban, A.J. Scholand, IDSim: An extensible framework for interoperable distributed simulation, in: The IEEE International Conference on Web Services, ICWS2004, 2004, pp. 532–539.

[11] http://www.cc.gatech.edu/computing/pads/tech-highperf.html, January 2007.

[12] IEEE 1516, IEEE Standard for High Level Architecture, 2001.

[13] IEEE1516.3, IEEE recommended practice for high level architecture (HLA) Federation development and execution process (FEDEP), 2003.

[14] K. Iskra, G. Albada, P. Sloot, Time warp cancellations optimisations on high latency networks, in: The 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications, DS-RT'03, 2003, pp. 128–137.

[15] K. Iskra, G. Albada, P. Sloot, Towards grid-aware time warp, Simulation: Transactions of The Society for Modeling and Simulation International 81 (4) (2005) 293–306.

[16] F. Kuhl, R. Weatherly, J. Dahmann, Creating Computer Simulation Systems: An Introduction to HLA, Prentice Hall, 1999, ISBN: 1-3-022511-8.

[17] M. Lees, B. Logan, J. Kings, HLA simulation of agent-based bacterial models, in: European Simulation Interoperability Workshop 2007, 07E-SIW-032, June 2007.

[18] N. Li, Z. Xiao, L. Xu, X. Peng, Research and realization of collaborative M&S services in simulation grid, in: The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, Singapore, May 2006.

[19] R. Minson, G. Theodoropoulos, Distributing *RePast* agent-based simulations with HLA, in: European Simulation Interoperability Workshop 2004, 04E-SIW-046, June 2004.

[20] R. Minson, G. Theodoropoulos, Distributing *RePast* agent-based simulations with HLA, Computation and Concurrency Practice and Experience Journal, Wiley, 26 pages [in press].

[21] M. Low, W. Cai, S. Zhou, A federated agent-based crowd simulation architecture, in: The 2007 European Conference on Modelling and Simulation, Prague, Czech Republic, June 2007, pp. 188–194.

[22] K.L. Morse, D. Drake, R.P.Z. Brunton, Web enabling an RTI-An XMSF profile, in: European Simulation Interoperability Workshop 2003, 03E-SIW-034, Stockholm, Sweden, June 2003.

[23] K.L. Morse, D. Drake, R.P.Z. Brunton, Web enabling HLA compliant simulations to support network centric applications, in: Command and Control Research and Technology Symposium 2004, 172, September 2004.

[24] H.F. Nielsen, E. Christensen, J. Farrell, Specification: WS-Attachments. http://www-106.ibm.com/developerworks/webservices/library/ws-attach.html, January 2007.

[25] C.M. Overstreet, R.E. Nance, O. Balci, Issues in enhancing model reuse, in: The First International Conference on Grand Challenges for Modeling and Simulation, San Antonio, TX, USA, January 2002.

[26] M.D. Petty, E.W. Weisel, R.R. Mielka, A formal approach to composability, in: The 2003 Interservice Industry Training, Simulation and Education Conference, December 2003, pp. 1763–1772.

[27] M.E. Pollack, M. Ringuette, Introducing the Tileworld: Experimentally evaluating agent architectures, in: National Conference on Artificial Intelligence, 1990, pp. 183–189.

[28] J.M. Pullen, R. Brunton, D. Brutzman, D. Drake, M. Hieb, K. Morse, A. Tolk, Using web services to integrate heterogeneous simulations in a grid environment, Future Generation Computer Systems 21 (1) (2005) 97–106.

[29] S.F. Railsback, S.L. Lytinen, S.K. Jackson, Agent-based simulation platforms: Review and development recommendations, Simulation: Transactions of The Society for Modeling and Simulation International 82 (9) (2006) 609–623.

[30] R. Fujimoto, J. Leonard II, Grand challenges in modeling and simulating urban transportation systems, in: The First International Conference on Grand Challenges for Modeling and Simulation, San Antonio, TX, USA, January 2002.

[31] K. Rycerz, M.T. Bubak, M. Malawski, P.M.A. Sloot, A framework for HLA-based interactive simulations on the grid, Simulation: Transactions of The Society for Modeling and Simulation International 81 (1) (2005) 67–76.

[32] B. Sotomayor, The Globus Toolkit 3 Programmer's Tutorial. http://gdp.globus.org/gt3-tutorial/, January 2007.

[33] S. van Splunter, N.J.E. Wijngaards, F.M.T. Brazier, D. Richards, Automated component-based configuration: Promises and fallacies, in: The Adaptive Agents and Multi-Agent Systems workshop at the AISB 2004 Symposium, 2004, pp. 130–135.

[34] G.K. Theodoropoulos, Y. Zhang, D. Chen, R. Minson, S.J. Turner, W. Cai, X. Yong, B. Logan, Large scale distributed simulation on the grid, in: The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, Singapore, May 2006.

[35] R. Tobias, C. Hofmann, Evaluation of free Java libraries for social-scientific agent-based simulation, Journal of Artificial Societies and Social Simulation 7 (1) (2004).

[36] S. Tuecke, K. Czajkowski, I. Foster, J. Rey, F. Steve, G. Carl, Grid service specification. http://www.globus.org/research/papers/gsspec.pdf, January 2007.

[37] A. Wytzisk, I. Simonis, U. Raape, Integration of HLA simulation models into a standardized web service world, in: European Simulation Interoperability Workshop 2003, 03E-SIW-019, June 2003.

[38] L. Xu, X. Peng, SSB: A grid-based infrastructure for HLA systems, in: The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, Singapore, May 2006.

[39] Y. Xie, Y.M. Teo, W. Cai, S.J. Turner, Service provisioning for HLA-based distributed simulation on the grid, in: The Nineteenth ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation, PADS 2005, Monterey, CA, USA, June 2005, pp. 282–291.

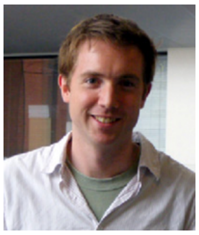
[40] S. Zhu, Z. Du, X. Chai, GDSA: A grid-based distributed simulation architecture, in: The Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops, Singapore, May 2006.

**Dr. Dan Chen** is currently a Postdoctoral Research Fellow with School of Computer Science at the University of Birmingham, United Kingdom. He was a Research Engineer at Singapore Institute of Manufacturing Technology before joining UoB in Aug 2004. He received a bachelor of science degree in Applied Physics from Wuhan University (China) in 1994; a master of engineering degree in Computer Science from Huazhong University of Science and Technology (China) in 1999. After that, he joined Nanyang Technological Univesity (Singapore) in 1999, and finished another M.Eng. programme and a Ph.D. programme from the School of Computer Engineering in 2001 and 2005 respectively. His recent research has focused on Grid computing, Distributed simulation and the development of simulation of Multi-Agent Systems.

**Dr. Georgios Theodoropoulos** is a Senior Lecturer in the School of Computer Science at the University of Birmingham (UK) where he has set up and leads the Distributed Systems Lab. He is also a founding Director of the Midlands e-Science Centre. He received his Diploma in Computer Engineering from the University of Patras (Greece) and his M.Sc and Ph.D. in Computer Science from the University of Manchester (UK). His current research is in parallel and distributed simulation, distributed virtual environments, Grid computing and Peer-to-Peer systems. He is currently Area Editor of the Simulation: Transactions of the Society for Computer Simulation International (SCS) and editorial board member of the International Journal of Simulation and Process Modelling.

**Dr. Stephen John Turner** joined Nanyang Technological University (Singapore) in 1999 and is Director of the Parallel and Distributed Computing Centre in the School of Computer Engineering. Previously, he was a Senior Lecturer in Computer Science at Exeter University (UK). He received his MA in Mathematics and Computer Science from Cambridge University (UK) and his M.Sc. and Ph.D. in Computer Science from Manchester University (UK). His current research interests include: parallel and distributed simulation, distributed virtual environments, grid computing and multi-agent systems. He is steering committee chair of the Principles of Advanced and Distributed Simulation conference and advisory committee member of the Distributed Simulation and Real Time Applications symposium.

**Dr. Wentong Cai** is an Associate Professor with the School of Computer Engineering at Nanyang Technological University (NTU), Singapore, and head of the Computer Science Division. He received his B.Sc. in Computer Science from Nankai University (P.R. China) and Ph.D., also in Computer Science, from University of Exeter (U.K.). He was a Post-doctoral Research Fellow at Queen's University (Canada) before joining NTU in February 1993. Dr. Cai's research interests include Parallel & Distributed Simulation, Grid & Cluster Computing, and Parallel & Distributed Programming Environments. His main areas of expertise are the design and analysis of scalable architecture, framework, and protocols to support parallel & distributed simulation, and the development of models and software tools for programming parallel/distributed systems. He has authored or co-authored over 180 journal and conference papers in the above areas. Dr. Cai is a member of the IEEE. He is currently an associate editor of *ACM Transactions on Modeling and Computer Simulation* (TOMACS), editorial board member of *Multiagents and Grid Systems — An International Journal*, and editorial board member of *International Journal of Computers and Applications*.



**Robert Minson** is a Research Fellow in the School of Computer Science at the University of Birmingham, UK. He received his M.Sc from the same institution in 2003. He currently works on large parallel simulation of Multi-Agent models as part of the MWGrid project at Birmingham. He is a member of the Distributed Systems Lab at Birmingham and has published previously in the fields of Parallel and Distributed Simulation, Multi-Agent Simulation, Interest Management in Large-Scale Virtual Environments, Massively Multi-player Computer Games and Peer-to-Peer Systems.

**Dr. Yi Zhang** works as a researcher at SAP Research CEC Belfast. Before joining SAP, he worked as a research fellow at School of Computer Science at the University of Birmingham. He received his B.Sc and M.Sc in Electrical Engineering from Xi'an Jiaotong University in 1992 and 1995 respectively. He received his Ph.D. degree in Computer Science from Chalmers University of Technology, Sweden in 2003. His research has focused on distributed computing, virtualization and performance engineering.