Grid-Enabled Software Conferencing for the SIP-RTI Runtime Infrastructure

by

Jin Kai Ren

A Thesis Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements For the Degree of

Master of Computer Science

Ottawa-Carleton Institute for Computer Science
Faculty of Science
School of Computer Science
Carleton University, Ottawa, Ontario

Your file Votre référence
ISBN: 978-0-494-33684-7
Our file Notre référence
ISBN: 978-0-494-33684-7

Abstract

The *High Level Architecture* (HLA) is a popular standard for distributed simulation interoperability. This specification defines the *Runtime Infrastructure* (RTI), which provides a set of services that support the simulations in carrying out these federate-to-federate interactions. To flexibly and reliably deploy and execute simulations on large-scale networks in wider simulation user community requires security and deployment management mechanisms in distributed simulation infrastructure like HLA.

The *Grid-enabled, Session Initiation Protocol (SIP) based RTI* is an extension of the *SIP-RTI*, which is a new model for RTI interoperability. However, SIP-RTI did not touch security issue, also it tightly coupled its components so that it is hard to manage deployment in large-scale distributed simulations.

The extensions are aimed at guaranteeing secure interoperation of large-scale distributed simulations and facilitating deployment management of components in SIP-RTI. SIP-RTI components are developing and deploying on top of Grid middleware (In this case, it is *Globus Toolkit 4*), and the communication in *Conferencing Infrastructure (CI)* is secured by exploring *Grid Security Infrastructure (GSI).* Meanwhile, *Grid Services* are used to decouple SIP-RTI components. So these components are portable and flexible to deploy.

Therefore, this Grid-enabled SIP-RTI provides not only a security solution to HLA, but also provides more flexible and portable interoperability solution.

Acknowledgements

I would like to thank Dr. Michael Weiss for his help and guidance with this project. His encouragement and advice were greatly appreciated. He is one of the best teachers and engineers I know and I am happy to have been his student during this research.

Dr. Trevor Pearce gave me important guidance to the initial concepts on which this research is based. I would like to thank him for this assistance. I would also like to thank my wife and my parents for their encouragement and support in pursuing my future. Without these people this work would not be possible.

Table Of Content

List of Figures

List of Abbreviations

A	- Available (floor)
API	- Application Programming Interface
BEEP	- Blocks Extensible Exchange Protocol
CallerID	- Caller Identifier
CDATA	- Character Data (used in XML)
CI	- Conferencing Infrastructure
ConfID	- Conference Identifier
CORBA	- Common Object Request Broker Architecture
DMSO	- Distributed COM (DCOM)
DoD	- Department of Defence
DTD	- Document Type Definition
FIFO	- First In First Out
FloorID	- Floor Identifier
FOM	- Federation Object Model
GSI	- Grid Security Infrastructure
GSS	- Generic Security Service
HLA	- High Level Architecture
HTTP	- Hypertext Transfer Protocol
IEEE	- Institute of Electrical and Electronics Engineers
IETF	- Internet Engineering Task Force
IP	- Internet Protocol
IPSec	- IP Security
ISO/OSI	- International Standards Organization/Open Systems Interconnection
JAIN	- Java API for Integrated Networks
LAN	- Local Area Network
LCC	- Grid-enabled Local Conferencing Component
LRC	- Grid-enabled Local RTI Component
LRC	- Local RTI Component
M&S	- Modelling and Simulation
MIME	- Multipurpose Internet Mail Extension
NIST	- National Institute of Standards and Technology
OGSA	- Open Grid Services Architecture
OGSI	- Open Grid Services Infrastructure
OMT	- Object Model Template
OOP	- Object Oriented Programming
ORB	- Object Request Broker

P2P	**- Peer-to-Peer**
PSTN	**- Public Switched Telephone Network**
QoS	**- Quality of Service**
RFC	**- Request For Comments**
RMI	**- Remote Method Invocation**
RPC	**- Remote Procedure Call**
RTI	**- Runtime Infrastructure**
SDP	**- Session Description Protocol**
SISO	**- Simulation Interoperability Standards Organization**
sLCC	**- SIP Local Conferencing Infrastructure Component**
sLRC	**- SIP Local RTI Component**
SOAP	**- Simple Object Access Protocol**
SCTP	**- Stream Control Transport Protocol**
TCP	**- Transmission Control Protocol**
TLS	**- Transport Layer Security**
UDP	**- User Datagram Protocol**
UML	**- Unified Modelling Language**
URI	**- Universal Resource Indicator**
VoIP	**- Voice over IP**
XML	**- Extensible Makeup Language**
XRTI	**- Extensible Run-Time Infrastructure**
XMSF	**- Extensible Modelling Simulation Framework**

Chapter 1 Introduction

The *High Level Architecture* (HLA) is a popular standard for distributed simulation interoperability. This specification defines the *Runtime Infrastructure* (RTI), which provides a set of services that support the simulations in carrying out these federate-to-federate interactions. To flexibly and reliably deploy and execute simulations on large-scale networks in wider simulation user community requires security and deployment management mechanisms in distributed simulation infrastructure like HLA.

The *Session Initiation Protocol (SIP) based RTI* (SIP-RTI) is a new model for RTI interoperability. However, SIP-RTI did not touch security issue, also it tightly coupled its components so that it is hard to manage deployment in large-scale distributed simulations.

This thesis describes Grid-enabled extensions designed for SIP-RTI [5] to resolve the security and deployment management issues in SIP-RTI. By developing and deploying SIP-RTI components on top of Grid Middleware, these extensions introduce strong security feature into HLA.

In this chapter, we present an overview of HLA and SIP-RTI, analyze its important problems; then present motivation for applying Grid technology to distributed conferencing system and HLA simulation; finally, outline the objective of the thesis, its scientific contribution and methodology. Our purpose is to explain the motivation and contribution of our research.

1.1 Overview of HLA Issues

The High Level Architecture (HLA) is essentially a set of rules that allow interoperability and reuse of simulations. Individual simulation, called federate, may be combined into a single, larger simulation, called a federation. Federates communicate with each other through the Run Time Infrastructure (RTI). The HLA rules require an object model that describes each simulation and federation, define federation-wide services, and specify the

interface between the simulations and RTI. Interactions between simulations in a federation execution are controlled by RTI.

HLA is intended to have wide applicability, across a full range of defense simulation applications including training, analysis, and engineering functions, at a variety of levels of resolution [1]. The various distributed simulations may be in various security domains, belong to various organizations and gain access to various RTI implementations for interoperation. Hence, there are two demands for HLA/RTI: one is RTIs' interoperability; the other is to secure network communication against unauthorized access in the federation.

However, the HLA specification has shortcoming for both needs. HLA defines RTI as an Application Programming Interface (API), not as an implementation. So as long as the interfaces and other HLA rules are maintained, the RTI may be implemented in any way. That flexibility results in that RTIs are generally not interoperable [2] due to differences in implementation. In addition, HLA includes no definition of security. The HLA *Management Object Model* (MOM), which provides an interface to the RTI for management tasks, provides the only scope for native security. The lack of working security solutions for HLA leads to a focus of international HLA researches.

1.2 Overview of SIP-RTI Issues

The SIP-RTI is a SIP-enabled RTI model. It focused on resolving the interoperability of HLA/RTI. In this model, a distributed *Conferencing Infrastructure (CI)* was designed and implemented underlying the RTI components. In essence, SIP-RTI transferred HLA semantics to software conferencing semantics. Its architecture (Figure 1-1) shows that

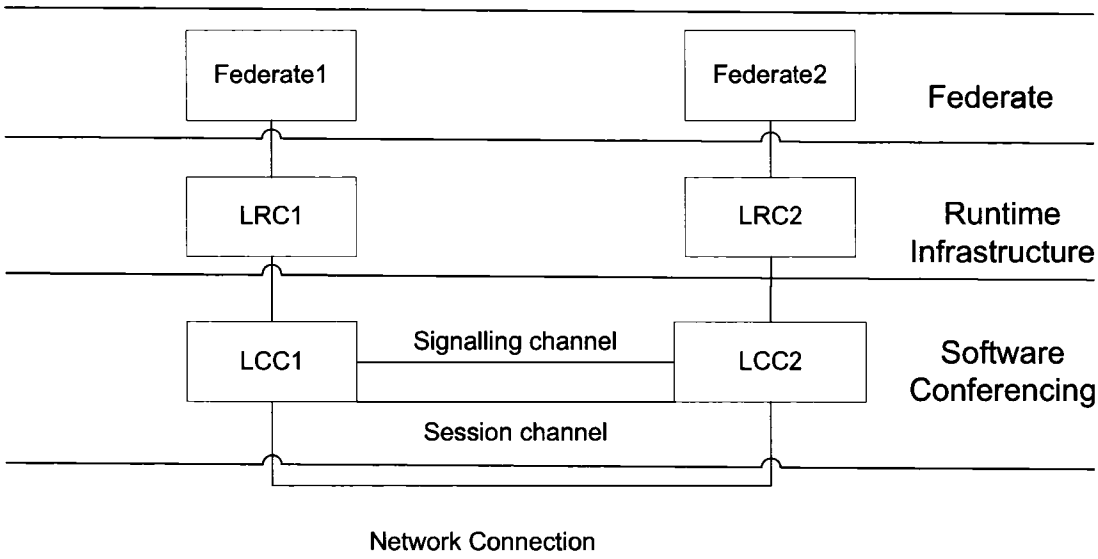

Figure 1-1 SIP-RTI Architecture

SIP-RTI comprises two components: *Local Conferencing Component (LCC)* and *Local RTI Component (LRC)*. LRC must makes use of LCC to interoperate with other LRCs. In CI, firstly SIP was used to establish signaling channel , then session channel was established through negotiation on signaling channel. However, this session channel is not secured. For large-scale distributed simulation, security is one of the primary concerns. Moreover, LCC and LRC are tightly coupled so that LCC and LRC must be deployed in the same process as federate. Additionally, only one LCC and LRC instance is allowed in each node. Therefore, it is hard to manage deployment of these components in large-scale distributed simulation.

1.3 Overview of Grid Computing

Grid computing is an emerging computing model that treats all resources as a collection of manageable entities with common interfaces to such functionality as lifetime management, discoverable properties and accessibility via open protocols. Grid technology has gained more and more attention from the commercial and scientific fields because of its capability in resource coordination, open standards and high efficiency.

The *Open Grid Services Architecture* (OGSA), developed by The Global Grid Forum [3], has already specified a set of standard interfaces for the most important services in Grid systems, including execution management services, information services, data management services and security services, etc.

The interest in Grid is growing as network speed is increasing more than computer speed [38]. As a newest distributed technology, The Grid is one of the most important concepts that strongly influence the area of scientific distributed computing. Since HLA is a well known standard in the area of distributed and interactive simulation, lots of effort is being devoted to take advantage of Grid for large-scale distributed simulation of HLA.

Globus Toolkit [4] is an open source software toolkit used for building Grid systems and applications. It is becoming the de facto standard middleware for Grid computing to create a more generic resource-sharing context.

1.4 Objective and Methodology of This Research

The thesis of this research is that a Grid-enabled SIP-RTI provides security and deployment management extensions to SIP-RTI [5], hence resolves both the security and interoperability issues in HLA/RTI. Meanwhile, a Grid-enabled software conferencing infrastructure will provide secured interoperability, not only of RTIs, but distributed system in general.

A fully distributed conferencing model called a Conferencing Infrastructure (CI), which was proposed in [5], has been refracted to service-oriented architecture, and deployed in Grid environment. We call it Grid-enabled software Conferencing Infrastructure. This CI implements a secured session layer communication system based on generic conferencing services and provides grid services to its user software. The CI first uses SIP to signal the session creation; then uses Generic Security Service API (GSSAPI) to create security context by making use of Grid Security Infrastructure (GSI); finally through the security context the TCP session channels were established to support communication among nodes participating in the CI.

The conference-based model (referred to as the crtiModel) of HLA semantics, which was proposed in [5] was also refracted and deployed in the Grid environment. We call it Grid-enabled crtiModel. This crtiModel is implemented as RTI API middleware that uses the underlying Grid-enabled CI's grid services and also provides grid services to CI for call back functions. Figure 1-2 depicts this layered components structure. As shown, when considered together, these two components result in a scalable and extensible RTI called the Grid-enabled SIP-RTI. The Grid-enabled SIP-RTI relies on the common underlying conferencing mechanism and the common security infrastructure –GSI to provides secure HLA services to federation running above it. Because the system is built on top of Grid middleware, it provides the RTI model with the benefits of Grid middleware components and services and may lead to the whole solution of RTIs' interoperability and security.

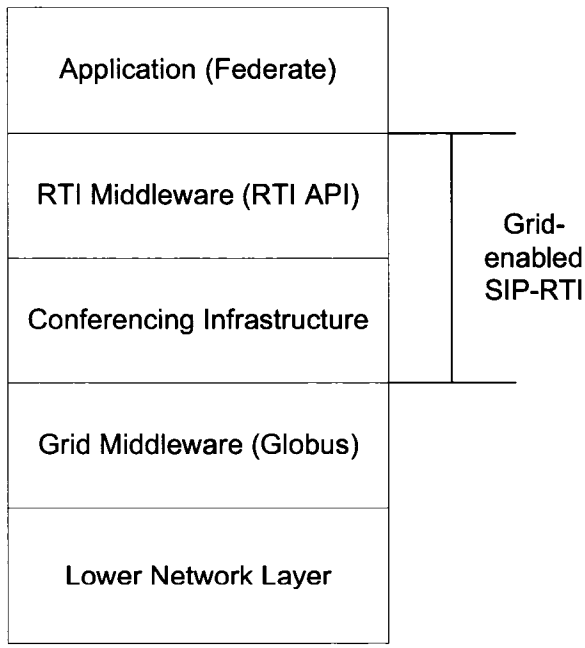

Figure 1-2 layered components structure for Grid-enabled SIP-RTI

1.5 Contributions

The Grid-enable SIP-RTI satisfies the thesis of this work due to the ability of the CI to provide communication among, not only RTIs, but any large-scale applications as well. In addition, because both CI and RTI components are deployed in Grid environment, they have a common sufficient security infrastructure –Grid Security Infrastructure.

The theoretical and technical contributions made by the work are described in the following.

- Contribution 1 – A services-oriented, fully distributed software conferencing architecture

The first contribution of the thesis is the design, development and implementation of an independent, portable software conferencing architecture. The services-oriented conferencing architecture provides more flexibility, scalability and portability to the conferencing system.

A significant characteristic of the grid-enabled CI is that it is totally decoupled from RTI components and so portable that it can locate anywhere in grid. The Index service built in Grid middleware can be used to dynamically discover LCCs.

Meanwhile, the resulting system has a real fully distributed multiparty topology. All nodes in CI are peers, thus they can carry out the same functions. The central conference node, called queen in [5], is removed. The CI's node lists are administrated in Index service, which is built in Grid middleware. Hence, dynamically discovering LCCs was enforced in a strongly secured way by secured grid services.

- Contribution 2 – Introduce strong security feature into HLA

Through deploying HLA components into grid, GSI, which is built in Grid middleware, provides an open standard security infrastructure to these distributed HLA components. The strongly secured session channels among CI nodes are achieved by exploring the GSI. Meanwhile, the interaction between LRC and LCC uses secured grid services,

which also based on GSI. In addition, deploying federate components in grid environment make it possible to be securely reused on different security level based on the user's grid security identity determined by trusted third party, such as *Certificate Authentication (CA).*

- Contribution 3 – Facilitate deployment management of SIP-RTI

By using grid services interface, LCC and LRC are decoupled. So they can be deployed in different processes. Moreover, Factory design pattern is used in these components, therefore, more than one LCC and LRC instances can be created in each node through factory service.

1.6 Organization of Thesis

The next chapter introduces the HLA, Grid technology and software conferencing system to the depth needed to understand this research. It also describes the SIP-RTI model provided by [5], because the Grid-enabled SIP-RTI is implemented based on that model. Chapter 3 reviews the literature of current RTI implementations and issues including interoperability and security. It also presents current work on extending HLA to Grid. Chapter 4 presents the motivation by analyzing the material presented in Chapter 3 and gives the architectural overview of the Grid-enabled CI and SIP_RTI. Chapter 5 describes the design and implementation of Grid-enabled software conferencing Infrastructure, and the design and implementation of Grid-enabled SIP-RTI and local Runtime Infrastructure Component. Chapter 6 presents a demonstration federation in detail to demonstrate our work satisfies our research goals. Finally, Chapter 7 presents conclusions and recommendations for future work.

Chapter 2 Background

This chapter provides an introduction to Grid Computing, HLA and software conferencing system that are relevant to the rest of this dissertation. The SIP-RTI implementation model provided in [5] is also presented. The research work in this thesis is an extension to the SIP-RTI implementation model. Understanding SIP-RTI is required to understand this thesis.

2.1 Grid Computing

Grid computing is an emerging distributed computing model that coordinates large-scale resources that have open standard interfaces and are distributed within dynamic, multi-institutional virtual organizations using standard, open, general-purpose protocols. A virtual organization (VO) refers to a set of individuals and/or institutions in which resource providers and consumers define clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs.

The widely recognized definition of Grid and its architecture was described in [6]. Figure 2-1 shows the four basic layers in Grid - they are fabric, connectivity, resource and collective.

- The fabric layer refers to the resources that are actually going to be shared in Grid system, for example operating system and local software.
- The connectivity layer is to enable resources to securely communicate. Network protocols, such as TCP/IP, HTTP, etc, and security infrastructure (GSI) [4] are in this layer.
- The resource layer refers to the services and protocols that are able to control the access and use of single resource. Some Grid middleware components, such as Grid Resource Allocation Management (GRAM) and efficient file transfer (GridFTP), are in this layer.
- The collective layer encompasses the services and protocols that are responsible for coordinate multiple resources. It contains ubiquitous infrastructure services for resource sharing, such as Index Service.

- The application layer refers to the applications that will be running on a Grid system. The applications are flexible to directly interact with any layer except for Fabric layer whenever required to.

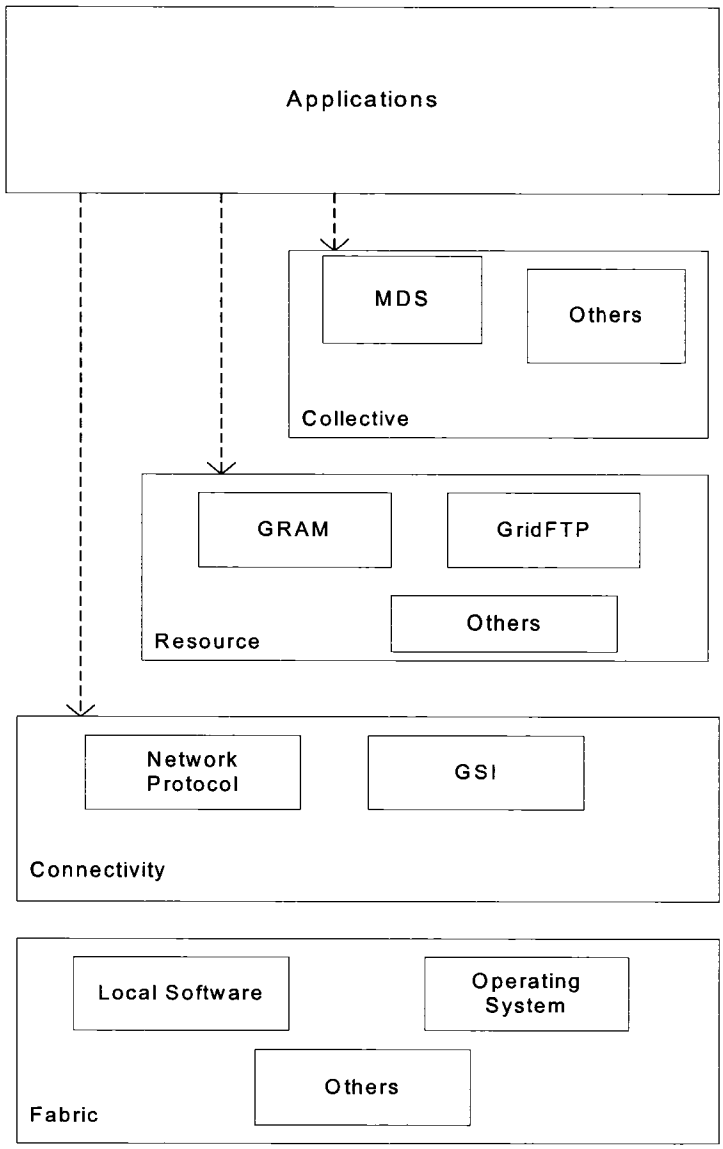

Figure 2-1 Grid Architecture

Based on this architecture, a grid system will usually consist of several different components and services. To allow intra-grid components and services interoperate correctly, the Open Grid Services Architecture (OGSA), developed by the Global Grid Forum [3], defines a common, standard, and open architecture for grid-based applications by specifying a set of standard interfaces for these services. To allow inter-grid interaction, Web services concepts [7] were adapted to Grid.

The Web services are another distributed computing technology (like CORBA, RMI, etc.). They have the following advantages over other technologies.

- Web services are platform-independent and language-independent because they use standard XML.
- Web services are ideal technology to build internet-scale application, because they use HTTP for transmitting messages. Most of the Internet's proxies and firewalls won't mess with HTTP traffic.
- Web services are ideal to build loosely coupled systems because they are message-oriented and rely on language-neutral XML to send messages and specify interfaces, etc.

Meanwhile, web services allow dynamical discovery of the published service functionality. Original Web services are stateless. For the same arguments they always return the same results. There are no internal states that can influence the answer.

The Open Grid Services Infrastructure (OGSI) [8] were the technical specification to specify how to implement the core Grid services as defined in OGSA . OGSI merged Web services and Grid concepts by introducing the concept of extending Web services to the transient and stateful Grid services with a notification mechanism.

Recently, the idea of Grid services has been changed to the Web Services Resource Framework (WSRF)[4]. The main idea remains the same, which is to converge concepts of Grid and Web services, but in a slightly different way. In order to add state to the Web service, instead of directly putting the state in web services, one can keep it in a separate entity called a resource, which will store all the state information. Each resource will

have a unique key as resource identifier. The pairing of a web service with a resource is called a WS-Resource [9]. So a stateful interaction with web services is an interaction with WS-Resource. WS-Addressing specifies Web-Resource-qualified *endpoint reference* (EPR)[10] that used to address a WS-Resource. EPR contains the URI of web services and resource identifier. Figure 2-2 shows the construct of EPR. The actual data items in WS-Resource are called the resource properties[37], which stands for the current state of the resource.

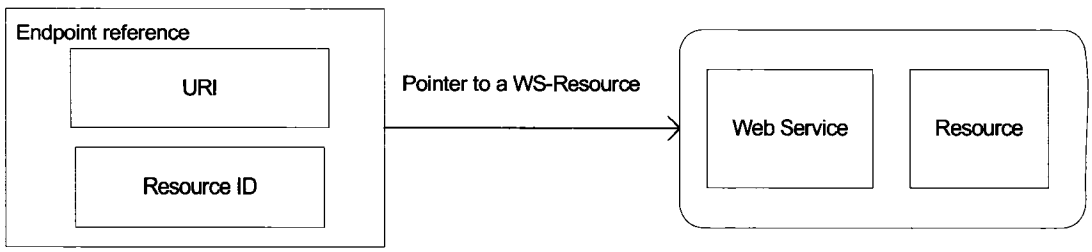

Figure 2-2 the Construct of EPR

In this way, the classical Web services can support features similar to those defined by OGSI. The concept of the upper level (OGSA) does not change. Figure 2-3 shows the WSRF-based Grid architecture.

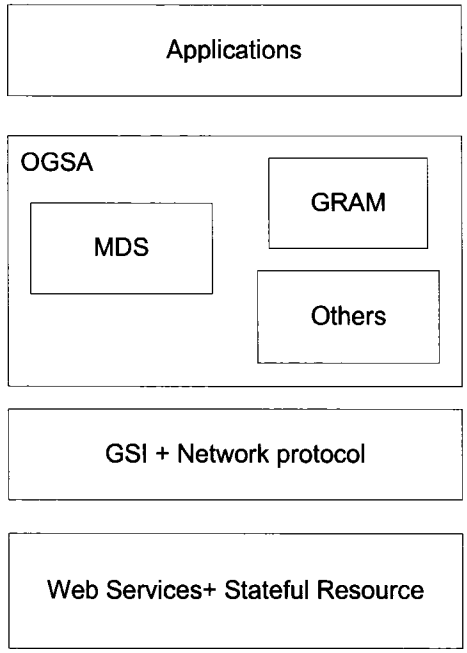

Figure 2-3 WSRF-based Grid Architecture

Globus Toolkit [3] is becoming the de facto standard middleware for Grid computing. It includes the high-level services defined in OGSA that we can use to build Grid applications. Its latest version, GT4, is the full-scale implementation of OGSA. It includes a complete implementation of the WSRF specifications and is a collection of loosely coupled components including security, Data Management, Execution Management, Information Services, and Common Runtime. The work of this thesis adopts GT4 by exploring its GSI and Information Services.

2.2 High Level Architecture

HLA defines software architecture. It is not a software implementation. HLA establishes common high-level simulation architecture to facilitate the interoperability of all types of simulations and models. HLA is designed to achieve standardization in the M&S community and to facilitate the reuse of M&S components.

HLA architecture implements an object-oriented network design. Each simulation system that is an interacting component of the HLA architecture is treated as an object and it is known as a federate. A collection of federates that participate in the same virtual environment is known as a federation. The federate registers its data members with software, Run-Time Infrastructure (RTI), which defines the HLA architecture. This software coordinates and controls the data transfer between federates. All federates send and receive their data through RTI. The RTI is responsible for handling low-level networking services, and for ensuring that the data are promptly sent from publishing federates to subscribing federates.

The HLA is defined by three components:

1. Object Model Template (OMT) [11]. It provides a common method for recording information, and establishes the format of Federation Object Model (FOM).

   All objects and interactions are defined according the standard OMT. The OMT provides a common framework for HLA object model information, and it promotes interoperability and reuse of simulations and its components. The FOM describes the object and interactions that are shared with other federates. It does not describe things internal to a single federate. The two main classes of OMT are Interaction Class and Object Class.

   Interaction Classes are comprised of parameters. They represent an occurrence or specific event in the simulation. These parameters are sent once through the RTI to other federates. The parameters do not persist after they have been received.

   Object Classes are comprised of attributes. They persist or have continued existence in the simulation. Federates update the state of an object instance by providing new values for its attributes.

2. HLA Rules [12]. These ensure proper interaction of simulations in a federation, and describe responsibilities of federates and their relationships with the RTI. The first five rules deal with federations, and the latter five with federates.

   2.1 Federation Rules

- Rule 1. Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the HLA Object Model Template (OMT).
- Rule 2. In a federation, all representations of objects in the FOM shall be in the federate, not in the run-time infrastructure (RTI).
- Rule 3. During a federation execution, all exchange of FOM data among federates shall occur via the RTI.
- Rule 4. During a federation execution, federates shall interact with the RTI in accordance with the HLA interface specification.
- Rule 5. During a federation execution, an attribute of an instance of an object shall be owned by only one federate at any given time.

2.2 Federate Rules

- Rule 6. Federates shall have an HLA Simulation Object Model (SOM), documented in accordance with the HLA Object Model Template (OMT).
- Rule 7. Federates shall be able to update and/or reflect any attributes of objects in their SOM and send and/or receive SOM object interactions externally, as specified in their SOM.
- Rule 8. Federates shall be able to transfer and /or accept ownership of attributes dynamically during a federation execution, as specified in their SOM.
- Rule 9. Federates shall be able to vary the conditions under which they provide updates of attributes of objects, as specified in their SOM.
- Rule 10. Federates shall be able to manage local time in a way which will allow them to coordinate data exchange with other members of a federation.

3. Interface Specification [13]. This identifies the callback functions each federate must provide, and defines the RTI services. The RTI is the implementation of HLA that provides network and simulation management services. There are six areas of services that handle specific tasks to be executed to manage the federation. Federates do not communicate with each other directly instead they invoke services through the RTI. These services are either RTI-initiated or federate-initiated services, and they reside in either the RTI or federate

respectively. The RTI presents an interface called the RTIambassador to each federate, and likewise each federate offers an interface called the FederateAmbassador to the RTI. The RTIambassador does not differentiate between federates and therefore it has a common interface. However each federate use those RTI services appropriate for its purpose and its FederateAmbassador may be unique. Each federate has a single point of contact with the RTI regardless of the number of processes or computers needed for the federate to execute its simulation. The six management services are designed to be independent and they can be used without referencing each other. They are summarized as follows:

- Federation Management. This area defines tasks that manage a federation execution. It includes tasks such as creating federations, joining federates to federations, observing federation-wide synchronization points, saving and restoring federation states, resigning federates from federations, and destroying federations.
- Declaration Management. This specifies data a federate will send and receive, and controls where the data is sent based on other federate interests. Its tasks include publishing objects or interactions that a federate intents to produce, subscription to specific data, and controlling the data flow by informing a federate whether other federates have subscribed to the data it intends to produce, so that it can stop producing the information when it is not needed.

- Object Management. This service is used to send and receive interactions, and register new instances of an object class and update its attributes. Its tasks include creating, modifying, and deleting objects and interactions, managing object identification, facilitating object registration and distribution, coordinating attribute updates among federates, and accommodating various transportation and time management schemes.

- Ownership Management. This service supports the sharing or transfer of ownership for individual object attributes, and it offers both push and pull

based transactions. The RTI allows federates to share the responsibility for updating and deleting object instances, however only one federate can update an attribute of an object at any given time. If the object is completely owned by one federate, then that federate is responsible for updating the attributes. An object instance may have various attributes owned by different federates however an attribute can only be owned by at most one federate. Moreover, only one federate has the privilege of delete the object instance.

- Time Management. This task management area controls the ordering of events in logical time advancement. The logical time does not represent any unit of real time. This service allows each federate to advance its logical time in coordination with other federates. It controls the delivery of time-stamped events so that each federate gets updated events. A federate can be either time-constrained or time-regulating or both or neither. In the time-constrained mode, the federate advance of logical time is constrained by other federates, and in the time-regulating mode the federate advance of logical time regulates other federates. The choice of time management is left to the federate's mode of operation.

- Data Distribution Management. This service supports efficient routing of data among federates, and it provides a flexible and extensive mechanism for isolating publication and subscription to regions of interest.

2.3 An abstract RTI Implementation Model

RTI is the software that implements the HLA interface specification. It provides an architectural foundation of common services to simulation systems and thereby promotes portability and interoperability. These services include construction and destruction of federations, support of object declaration and management between federates, providing communications between federates, and managing federation time. RTI implementations often use a single, global process to support federates joining and leaving a federation execution, and to facilitate data exchange between participating federates. This process is

called the FedExec. The FedExec is created by the first federate that successfully invokes the Create Federation Executive service of a particular federation execution.

The abstract representation is based on the use of LRCs, because in most HLA RTI designs individual nodes typically access the RTI through a locally instantiated RTI component. This generic model is shown in figure 2-4. Federates access HLA services using the RTI API, viewing the RTI as a black box. A federate has no knowledge of the how the RTI is implemented other than the services it offers. These services are made available to a federate via an RTIambassador that, in the model, is considered part of the LRC. The RTI is composed of one or more LRCs, which may be distributed. An LRC may support one or more federates. The model supports a variety of implementations.

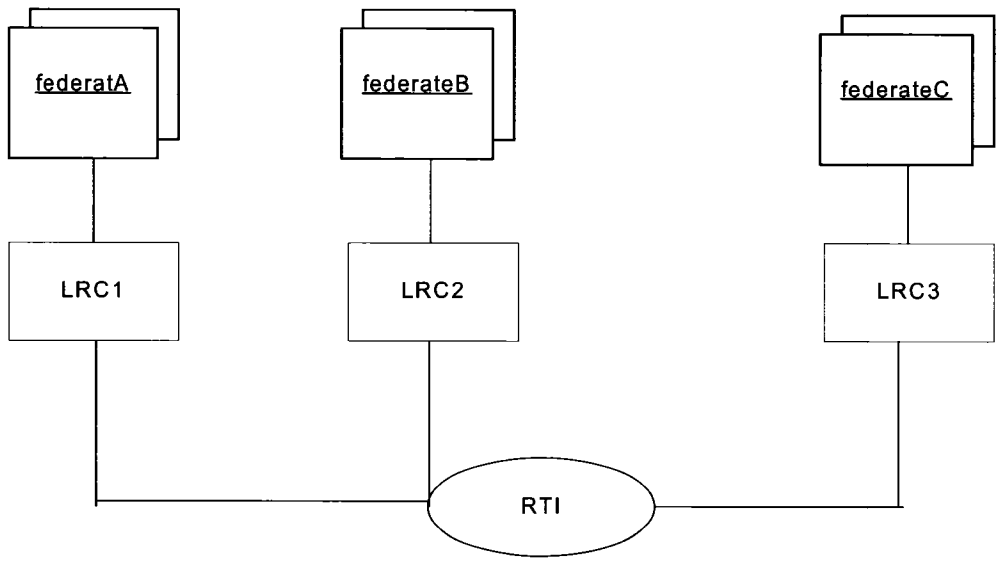

Figure 2-4 RTI generic model

2.4 Overview of Software Conferencing

A *conference* is a set of members or participants that share a common interest [5]. Conferencing includes two main components. First, participants must be able to join and leave a conference and their privileges are defined. The second component, conference control, can be considered as consisting of three parts: creating, modifying and deleting conferences as a whole, user management (adding and deleting conference participants and modifying their conference privileges), and resource contention management, also known as floor control [27].

There is lots of work, particularly in SIP communities, studying the challenges of implementing network-based conferencing. These researches focus on the development of conferencing frameworks and include research into architectures and conference control mechanisms. The SIP was, in fact, originally developed as a large scale, multiparty conferencing protocol [28] and, therefore, is involved in much of this research. Much of this research is being done by various working groups of the Internet Engineering Task Force (IETF) [29] and takes the form of Request For Comments (RFC) or Internet-Draft documents.

Six various conferencing models supported by SIP are defined by both the architecture and control methodology used in [30]. [31][32] Generalize these models to three: the multicast that relate directly to the loosely coupled model; the mixing model that relate directly to the tightly coupled models; and the more recent occurrence of conferencing model---fully distributed multiparty conferencing.

However, most of the earlier work discusses only the floor control aspects of conference control. Furthermore, some papers focus on multicast conferences [33,34,35]; in the absence of a widely available Internet multicast service, we will focus on the currently more common conferencing architecture. Moreover, current work generally targets the human conferencing problem. In [5], the authors provide a prototype of software conferencing system that aims to provide protocol-agnostic, universal framework and related components.

A software conference is a set of software applications that collaboratively work for share a common interest. Unlike traditional conferencing, the conference stakeholders are software application, not people. However, they have the same semantics. Like traditional conferencing, software conferencing includes two main components: conference setup and conference control.

Software conferencing is to provide communication infrastructure for the interoperability among different software applications that are potentially in different domain. Software applications are diverse, they may locate in different security domain, and have different implementation platform.

This thesis is to provide a general conferencing solution for large-scale software applications. To meet this goal, the conferencing system should be portable, scalable, easy to extend, generic, reliable and secure. The portability and scalability requirement means that the conference system can be independently deployed in any platform and support reasonably large, possibly geographically distributed, conferences. The security requirement means that only the authenticated members can do their authorized operations, and the communications inside conference caused by these operations must be secured.

2.5 SIP-RTI Model

This thesis is an extension and refashion on the SIP-RTI implementation model [5]. Understanding of this model is required to understand this thesis.

The SIP-RTI is a SIP-enabled RTI. It was motivated to resolve the interoperability of any set of distributed applications. By revealing the fundamental reason of interoperability, [5] point out that the interoperability issue is caused by the reason that an explicit session layer is missing from the HLA specification. Hence, a distributed conferencing infrastructure is designed and implemented to provide a common communication

mechanism at session layer. Then HLA semantics were represented on the public conferencing semantics.

Two important works had been done in this model. One is that a prototype of distributed software conferencing infrastructure (CI) was designed and implemented. The other is the transition from HLA semantics to software conferencing semantics on API level, which is using conferences to represent the federation, object classes and attributes, using speaker and listener attendees to represent federates and their interests, using floors to represent object class instances, and using conferencing services to represent the HLA RTI services.

The SIP-RTI comprises two components as shown in Figure 1-1. The first is the SIP-based, general-purpose, distributed, XML message-oriented software conferencing system called the Conferencing Infrastructure or CI.

The SIP Local Conferencing Infrastructure Component (sLCC) implements the local component of the CI and maintains both its own local state, and the global state information that is of importance to the conference. Each node participating in the CI supports a single sLCC. Each sLCC is part of the overall CI and used to maintain the state information of CI, which stand for the federation interaction in SIP-RTI. For communication among sLCCs, one special sLCC, named qLCC, have to be created at the beginning to hold the sLCCs list, which is a list of the host name of the CI participants. All other participating sLCCs first acquire the participating sLCCs list by establish session channel with qLCC using SIP signaling channel, then establish session channels with any other participating sLCCs through through negotiation in signaling channel created by SIP. These session channels are common TCP socket based connections. All sLCCs communicate by passing XML-format message in the session channels. As a result, the nodes in CI create a mesh network. Meanwhile, all participating sLCCs are peers in conference message communication. Each node can act as a server or a client as required. The CI provide conferencing services API to user applications, including the following services:

1) create and destroy conferences,

2) place and remove attendees in/from a given conference,

3) place and remove floors in/from a given conference,

4) maintain identification of a floor's owner,

5) support floor status changes (ownership transfers), and

6) make and deliver announcements made in a conference to all affected listeners.

Each sLCC in the same federation has the same conference map, which is the structure of parent and child sub-conferences based on the FOM, which is a XML file provided by federate. The conference map is established in each sLCC by the sLRC (using the sLCC API' s conference creation service) before the sLCC is joined to the CI and before any further API activity takes place. The CI also provides a way to control call synchronization and concurrency in one sLCC and distribute data in sLCCs to maintenance consistent global state. The detail can refer to [5].

The second component is Local RTI Component (sLRC), a custom-built middleware package designed to use the services of the CI to create an IEEE 1516 RTI. sLRC uses the services provided by its underlying sLCC to establish, manipulate and maintain conference state to achieve the transition mapping from HLA RTI services to software conferencing services. In essence, the FedExec was moved to underlying CI. Figure 2-5 depicts some services mapping between RTI and CI.

In fact, the sLRC is an intermediary between the federates it supports and the sLCC. Therefore, the sLRC has to control the mapping of three types of designators. Figure 2-6 shows the designator mapping lists. The object classes and their attributes defined in FOM are mapped to corresponding handles, which are used by federates to make calls on sLRC, which translate them to the equivalent sLCC conference ID in CI. The interaction

HLA services	Conferencing services
createFederationExecution	createConference
publishObjectClassAttribute	addAttendee
registerObjectInstance	addFloor
updateAttributeValues	makeAnAnnouncement

Figure 2-5 Services mapping between RTI and CI.

classes are mapped to the ObjectInstanceHandle in sLRC, which in turn mapped them to floor ID in sLCC. Similarly, federate itself is mapped to FederateHandle in RTI, which in turn mapped it to Caller ID in sLCC.

Element in FOM	LRC Handle	LCC Identifier
ObjectClassName	ObjectClassHandle/AttributeHandle	ConfID
N/A	ObjectInstanceHandle	FloorID
N/A	FederateHandle	CallerID

Figure 2-6 Designator Mapping Lists Maintained by the LRC [5]

To establish a SIP-RTI node an sLRC is first instantiated by a federate. When the first call to the HLA, "Create Federation" service is made on an sLRC it instantiates the node's sLCC. The sLRC then uses the sLCC's services to create a local copy of the

conference hierarchy, called a conference map, which is used in the CI in accordance with the FOM. The new sLCC then joins the CI by using the SIP to create signaling channels, which are then used to negotiate the setup of session channels with all participating sLCCs.

2.6 Summary

This chapter presents various concepts related to our research. We start with an overview of Grid computing, which can establish a dynamical large-scale distributed computing environment. Next we present the specification of HLA, which is the well known standard of distributed interactive simulations. Then we present the new evolving concept—software conferencing, which is mostly derived from traditional SIP conferencing. To facilitate the understanding of our work, we also briefly describe the HLA RTI implementation model, especially the new model – SIP-RTI, which integrates software conferencing into RTI to make use of conferencing semantics to address RTIs' interoperability issue.

The description of these concepts presented in this chapter allows us to find out that both HLA/RTI and software conferencing system are possible to take advantage of Grid technology for large-scale distributed application. This is the general derivation of our research.

Chapter 3 Literature Review

HLA is intended to have wide applicability, across the full range of defense simulation applications including training, analysis, and engineering functions, at a variety of levels of resolution [14]. The large-scale distributed simulations may be in various security domains, belong to various organizations and gain access to various RTI implementations for interoperation on demand. Hence, there are three demands for HLA/RTI: one is RTIs' interoperability; one is to secure network communication against unauthorized access among the federation, the third is dynamical discovery of federation for providing RTI services on demand.

However, the HLA specification has shortcoming for all the three needs. HLA defines RTI as an Application Programming Interface (API), not as an implementation. So as long as the interfaces and other HLA rules are maintained, the RTI may be implemented in any way. That flexibility results in that RTIs are generally not interoperable [2] due to differences in implementation. In addition, HLA includes no definition of security. The HLA Management Object Model (MOM), which provides an interface to the RTI for management tasks, provides the only scope for native security. Finally, HLA also does not support dynamical discovery of federation. To discover federation, a static configuration file that explicitly specifies the endpoint of RTIExec is needed.

There is lots of work on the "provision of services over and above what is available from the RTI"[15]. Most of them are focus on RTIs' interoperability, security of federation, and dynamical discovery of federation. Aiming at various reasons that cause these two issues, many solutions were presented.

3.1 RTI Interoperability

For interoperability, current solutions focused on the cause of RTIs' various implementation languages or platform, various implementation algorithms and both. [16] proposed an approach using the Common Object Request Broker Architecture(CORBA) technology based on the its programming language portability and flexibility. In this

effort, the interoperability of RTI could be achieved across various platforms. [2][17] proposed and presented an approach based on a wire standard. A wire standard forces system components to use the same algorithms by defining the syntax and semantics of each inter-component message in a distributed system. This approach makes up the issue caused by various algorithms implementation. Additionally, Extensible Modeling Simulation Framework (XMSF) [18,19] is used to provide solution aiming to both causes. XMSF uses Web Services as distributed technology to integrate various platforms. And use the Blocks Extensible Exchange Protocol (BEEP) [20] to provide a common messaging protocol. The BEEP also allows the definition of security services, thus, this approach could be a potential solution to RTI's interoperability and security issues.

To reveal the fundamental reason of interoperability, SIP-RTI model was provided in [5]. [5] point out that the interoperability issue is caused by the reason that an explicit session layer is missing from the HLA specification. Hence, a distributed conferencing infrastructure is provided as a common communication mechanism at session layer. Then HLA semantics were represented on the public conferencing semantics. Once again, this thesis is tightly based on the SIP-RTI implementation model. Understanding of this model is required to deeply understand this thesis. The detail SIP-RTI is described in section 2.4.

3.2 RTI Security

HLA includes no definition of security. The HLA Management Object Model(MOM), which provides an interface to the RTI for management tasks, provides the only scope for native security. Compared with the solutions on the interoperability issue of RTI, Currently there is no easy solution to security in HLA. [21][22] identified all valid requirements for security provisions: Entity registration, Access control mechanism, Entity authentication, Secure communication. To satisfy both the security requirement above and HLA rule, [23] proposed three security architectures: single security level, multiple single security levels and multiple MLS security domains. [24] presented the security extensions to ONERA/CERT implementation of RTI. It aimed to guarantees

secure interoperation of simulations belonging to various mutually suspicious organizations. In this implementation, a trust third party (TTP) operates a shared LAN where companies are free to connect machines hosting their federates. Each machine is allowed to host federates from only one company. All communications between company machines are mediated and authorized by the RTIG [25]. It also extended RTI services to add security domain filters for publication and subscription. RTIG transmits the UpdateAttributeValue messages only to authorized subscribers. Secure Medium Access Control (SMAC) protocol [26] was used to secure the communication between RTI Ambassador process (RTIA) and RTIG process. However, this solution focused on adding security mechanism to implementation of ONERA HLA RTI. So it subjected to the limitation of existing architecture of ONERA HLA/RTI prototype.

[21] proposed an approach to implement RTI using IP Security Protocol (IPSec) in conjunction with public key infrastructure(PKI), which could provide the basis for a secure implementation for HLA/RTI. In this solution, security services were provided at the IP layer, and IPSec policy must be applied to any machines involved in federations. In essence, this solution is to provide a common security infrastructure to all components in federation. [21] also demonstrates this approach can be applied to three security architectures for three security objectives. All the security objectives will be discussed in Chapter 4.

3.3 Discovery of Federation

HLA also has another important issue, which is that HLA does not support the discovery of federation services (such as the FedExec) and/or federation participants.

[43] noticed that current solutions in HLA implementations do not perform efficiently in Wide Area Networks and that traditional approaches to high-performance RTI implementations assume relatively static configurations of federates which are not sufficient for the highly dynamic nature of large-scale distributed simulation.

In the DMSO RTI [44], to discover the control process RTIExec, the endpoint of RTIExec has to be explicitly specified in the configuration file. This static nature lacks flexibility. Federates have to be mapped to computing resources before the simulation starts, HLA does not support dynamic allocation.

Moreover, In order to run a distributed simulation over the Wide-Area-Network (WAN) using the HLA RTI directly, special arrangements have to be made beforehand to ensure of the availabilities of the required hardware and software. Such arrangements are typically made with a centralized control or simply within an organization, because inter-organizational sharing of resources involves issues such as security. With the increase in the scale and complexity of simulations, large amounts of resources are required, and provisioning of services becomes more and more difficult. In the case of HLA-based distributed simulation, the resources of RTI, the simulation model, and /or the underlying infrastructure may not be available when they are needed. Therefore, The dynamical discovery of HLA federates and RTI is another issue for RTIs.

3.4 Grid Solutions for large-scale HLA

In section 2.1, we have briefly presented Grid concepts and evolution of its architecture and technology. The Grid is one of the most important concepts that strongly influence the area of scientific distributed computing. One of the most important Grid technologies is the Globus Toolkit [4]. Globus included a tool for job submission called Grid Resource Allocation Management (GRAM), efficient file transfer (GridFTP), Grid Security Infrastructure (GSI) and Monitoring and Discovering Service (MDS). To achieve interoperability between distributed systems, grid services are defined as the standard "InterGrid" protocols.

The Grid is a promising approach for various applications, including not only data intensive applications and the projects that aim to develop Grid infrastructure for computing, but also projects using Grid for distributed simulations. Since HLA is a well-known standard in the area of distributed and interactive simulation, lots of effort is being

devoted to take advantage of Grid for large-scale distributed simulation of HLA. In this section, we analysis the work related to merge HLA and Grid concepts.

To address dynamic discovery of HLA federation, [41] presented a framework using Grid services for executing large-scale HLA-based distributed simulations. Figure 3-1 shows architecture of the framework. The approach assumes that each federate is a simulation model encapsulated in a Grid service that is registered in an Index Service together with the RTIExec Service. By querying the Index Service, the client first retrieve the multicast endpoint used by the RTIExec and the handle of Federate Factory Service, and requests Federate Service instance be created. Then the client instruct the Federate Service to create simulation federates for the federation with the multicast endpoint passed through as a parameter. The created federates proceed to connect to the RTIExec process identified by the multicast endpoint and carry out the actual simulation. The simulation is set up and runs using HLA RTI communication. Therefore, this framework provides great flexibility for dynamical discovery of federation.

To support federation discovery and security of the federate logic, [42] provides a distributed simulation framework, called HLAGrid. This framework uses a Federate-Prox-RTI architecuture, which allows resources on the Grid to be utilized on demand by using Grid services. Figure 3-2 shows the architecture of this framework. The architecture hides the heterogeneity of the federates, federates' execution platforms, and how the federates communicate with the RTI.

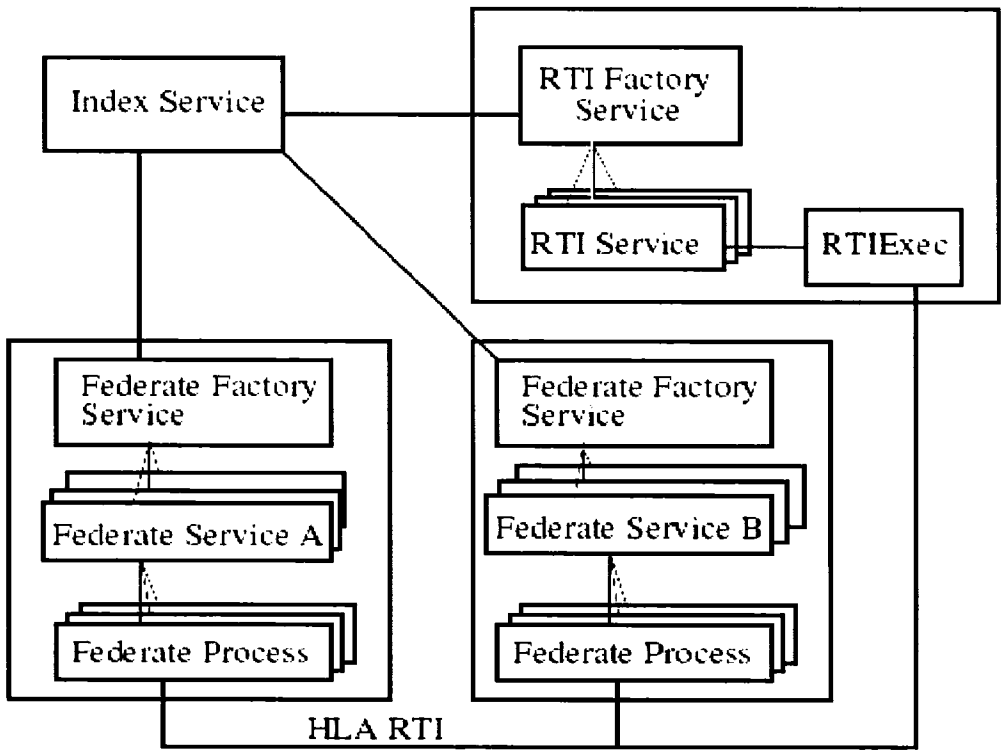

Figure 3-1 Architecture of a framework for large-scale distributed simulations with Grid services

The HLAGrid framework also has an Index Service, which is needed for maintaining the mapping between federations and handles of corresponding RTI services instances. Thus the dynamical discovery of federation is supported.

To support the security of the federate logic, a new entity, proxy, is introduced to act on behalf of the clients' federate code to communicate with proxies of other clients through the RTI. Proxies are executed at remote grid resources. Federate and their respective proxies communicate with each other through Grid services, and a Grid-enabled HLA API, which provides the standard HLA API to the federate, is implemented to translate the communications into Grid services invocations. When federates use standard HLA

APIs to make calls to RTI, the Grid-enabled HLA APIs translate them to Grid services requests to remote proxies. The proxies are responsible for translating Grid service invocations into normal federate initiated RTI invocations.

Therefore, federates can be run on any type of machine architecture, and the communication over the Grid network are hidden from federates by the Grid-enabled HLA APIs' translation. Moreover, Grid services invocation provides more secure, scalable and coordinated management.

However, the framework only use Grid services to secure the communication between federates and proxies, it did not mention any approach to secure the communication between RTIs.

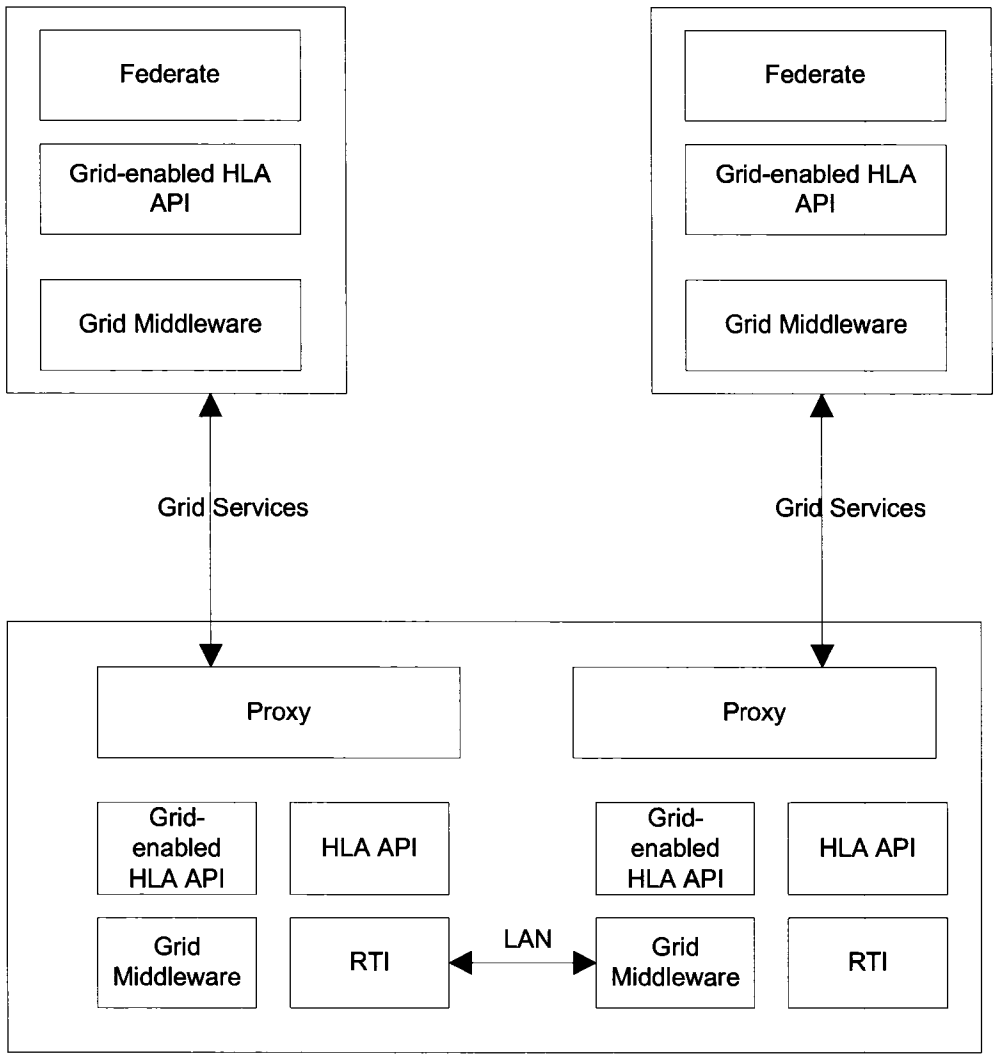

Figure 3-2 Architecture of Proxy-based HLA simulation on the Grid

3.5 Summary

This section presented three important issues of HLA federation: RTIs' interoperability, RTI security and dynamical discovery of federation. We started with the need of these three properties for large-scale distributed simulation. Next we stated the shortcoming issues of HLA for the three needs. Then for each of the issues, we described and analyzed the related effort that was trying to resolve it. Finally, we presented the Grid solutions that take advantage of Grid to HLA.

The analysis presented in this chapter allows us to find out that there are no a general solution that can address all the HLA issues at the same time without Grid technology. Also, we can find that currently most effort that merges Grid and HLA is only focus on the discovery of federation, interoperability between different federate using different languages and platforms, and security between federate and RTI component. In this research, we mostly address security issues of HLA/RTI by take advantage of Grid on HLA and software conferencing system. Figure 3-3 compares the two Grid solutions with Grid-enabled SIP-RTI.

	Interoperability	Security	Discovery of Federation
[41]	N/A	N/A	Yes
[42]	Support interoperability between different federates	Support security between federates and LRC	Yes
Grid-enabled SIP-RTI	Support RTIs interoperability by CI	Support security between LRCs	Yes

Figure 3-3 Comparisons Between Grid-enabled SIP-RTI and Other Grid Solutions

Chapter 4 Architecture

Security, interoperability and discovery of federation issues are the most important issues with HLA. All of them have received lots of international researches as mentioned in Chapter 3. This chapter first presents the motivation of this research through an analysis of previous researches about HLA/RTI problems, and conferencing system in SIP-RTI in [5]. Then introduces the overview of architecture of Grid-enabled software conferencing infrastructure and Grid-enabled SIP-RTI.

4.1. Motivation

The basic idea comes from the perspective that each LCC and LRC is a WS-Resource serving HLA federation in grid by a range of clearly defined collaborative problem-solving strategies. So that it can achieve large-scale distributed simulations in VO.

4.1.1 Motivation for HLA Security

Although lots of researches had been carried out, currently there is no easy solution to security in HLA. [21][22] identified all valid requirements for security provisions as Entity registration, Access control mechanism, Entity authentication, Secure communication and depicted four level security objectives.

The first level is to satisfy the first level HLA security requirement, that is Federations operating at one security level. All components in HLA are in the same security level , thus supposed to handle data at a unique security level. The key property of this level security objective is the users of federates are supposed to be mutual trusted, and all data transferred through federation should be secured, only the resources providing valid credential can involve the federation work flow.

The second level, multiple single security level, is to satisfy the second HLA security requirement, which is simulations within federations can operate at different security levels. It aims to secure interoperation of simulations belonging to various mutually

suspicious organizations [24]. In this level, the federation objects are regarded as confidential or public depending on the simulation organization's demanding. Only public data can be transferred to federates belonging to other simulation organizations, while confidential data can be exchanged only to federates belonging to the same organization. [24] provided an example implementation to achieve this objective. In [24], different organizations have separate federations with the same FOM. Each federation has one RTI center process, which communicates with RTI ambassador located on simulation machine. SMAC (Secure Medium Access Control) is used to secure communications between local RTI ambassador and RTI center process, A RTI filter is provided to sort out data's security classification to prevent confidential message from transferring to RTI center process belonging to others.

The third level, multiple MLS (Multi-Level Secure) security domain, is to satisfy the third HLA security requirement, which is the objects and their attributes in FOM transferring among simulations can operate at different security levels. [36] proposed an example security architecture for this objective. In [36], federates have to provide an XML Security Rule Definitions (SRD) to SexProxy nodes which have Security Rule Storage (SRS). In SRD, each federation object or its attributes in FOM can be defined various views for various federates. SexProxy can identify various federates through authentication, then control the transferring the view of various federate object to corresponding federate according the object's security rule definition. So the security can be fine-grained to attribute level.

The fourth level is to satisfy the fourth HLA security requirement, which is to reuse of simulations at different security levels. Any federation participant user with their own privileges can reuse simulations. Hence, different user may have different security level for the same simulation. So the identity of simulation varies from various users. [22] states that requesting participant of a federation requires to be verified by authority to make sure that they are authorized to participate in the federation

Although the last three levels might be inconsistent with the HLA rule that states that every federate that subscribed to a class of the FOM should receive all the information on this class. There is large security demanding to widely application of distributed simulation.

However, current solutions cannot satisfy the entire security requirement for worldwide large-scale distributed simulation. The fundamental problem is the lack of a common open standard security infrastructure under the distributed interactive components in HLA.

[21] proposed an security approach for the first three security objectives. It provides RTIs a common security infrastructure using Internet Protocol Security(IPSec) and public key infrastructure(PKI). IPSec is used to secure communication channels, and PKI is used to provide identity to objects and their attributes in FOM for implementing various security levels in one FOM. However, there are two limitations for this solution. First, IPSec depends heavily on physical security to ensure no one can gain control of the host implementing IPSec. Second, IPSec is implemented using host level authentication and encryption. It is no way to satisfy the four level security objective that allows reusing of simulation at various security levels. Because this objective implies user level authentication and encryption

Grid Security Infrastructure (GSI) provides a open standard security mechanism integrated with Public Key Infrastructure (PKI). All HLA components deployed in Grid could make use of GSI to achieve all the security objectives over the large-scale distributed simulation. GSI provides a more flexible security mechanism without the limitations in [12]. At the same time; it provides all the abilities that IPSec can provide.

4.1.2 Motivation for Deployment Management of SIP-RTI

Many solutions as mention in Chapter 3 were provided focusing on the various platforms, various implementation algorithms or both of the RTI implementations. Through revealing the missing of an explicit session layer from HLA specification, [5] stated that

the ideal solution should provide a common communication mechanism, implemented via a distinct session layer and based on the semantics of a publicly specified model. The definition of a public model will require all applications that adhere to it be designed and implemented in accordance with common semantics thus promoting interoperation. Based on this principle, a SIP-RTI implementation model was presented as described in Chapter 3.

However, there are at lease two further works needed to be done. The first one is the work on security. Because the CI component in SIP-RTI is a fully distributed model, security is one of the important requirements. However, it did not touch it at all. The second is the work for other concerns, such as portability and flexibility. All the components including CI component, RTI component, federate component are tightly coupled. An LRC must be instantiated by a federate. Then an LCC must be instantiated by the LRC. Therefore, these components are not independent. Another important problem is this SIP-RTI is not suitable for large-scale simulations because of two reasons. The first reason is that the CI in SIP-RTI is a full-mesh conferencing model that is only suitable to small-scale conference [31]. If this model is used in large-scale simulation in which lots of graphically distributed simulation involved, the CI would be also be in large scale, the performance would be a disaster. The other reason is that the LCC coupling with LRC enforce high workload to each node, which in turn raises the requirement of machine's hardware capacity.

Grid technology not only provides an open standard GSI, which can be explored to resolve the security issue in HLA, but also benefit creating flexible, resilient operational infrastructure, improving optimal utilization of computing capabilities based on an open set of standards and protocols that enable communication across heterogeneous, geographically dispersed environments. With grid, computing and data resources can be pooled optimized for large capacity workloads, shared across networks and collaborated.

4.2 The Architecture of Grid-enabled Software Conferencing Infrastructure

The Grid-enabled software conferencing infrastructure is a fully distributed, services-oriented system deployed in grid. It provides conferencing services with secure grid services interface to general software applications for their interaction, and strongly secure the conference message communication in their SIP session channels.

In this CI, LCC instances not only can be dynamically created in any grid node by grid administrator, but also are allowed more than one instance in any node. Therefore, LCCs are totally independent and portable. Conferences are encapsulated within LCC that are created through LCC Factory Services, and a central Index Service is responsible for maintaining participating list of LCCs. Another grid service, LCC Instance Service is to provide conferencing services to user software. In this way, Grid services can be dynamically mapped to LCC resources that are diversely distributed and hidden from users. Dynamic discovery of LCCs is fully supported, too. With this approach, LCC resources management is provided. Moreover, by exploring GSI, Generic Security Services (GSS) was used to establish security context, which in turn establish the secured session channels among LCCs.

The proposed architecture of the software conferencing system over Grid is illustrated in Figure 4-1. It is a WSRF [9] compliant implementation that use factory/instance pattern. It consists of several key Grid services, namely the LCC (Local Conference Component) Factory Service and the LCC service, which are all designed and implemented based on GT4. The Index Service acts as an information server to facilitate dynamic discovery of the LCC resources that include conference objects.

- Index Service: The main function of the Index Service is to keep LCC's information, including the endpoint reference of LCC resource which stand for the location of LCC, and related Resource Properties [19], such as conference subject which LCC is working for, LCCs' global security principals; to support queries from client application; and to support status update of resource properties caused by LCC

Service. Hence, it provides operations for query and operations for register and update LCCs' resource properties. The Index Service has one Index entry for each registered LCC. With the Index Service, different LCC resources can be located dynamically so that client applications can resolve to it directly.

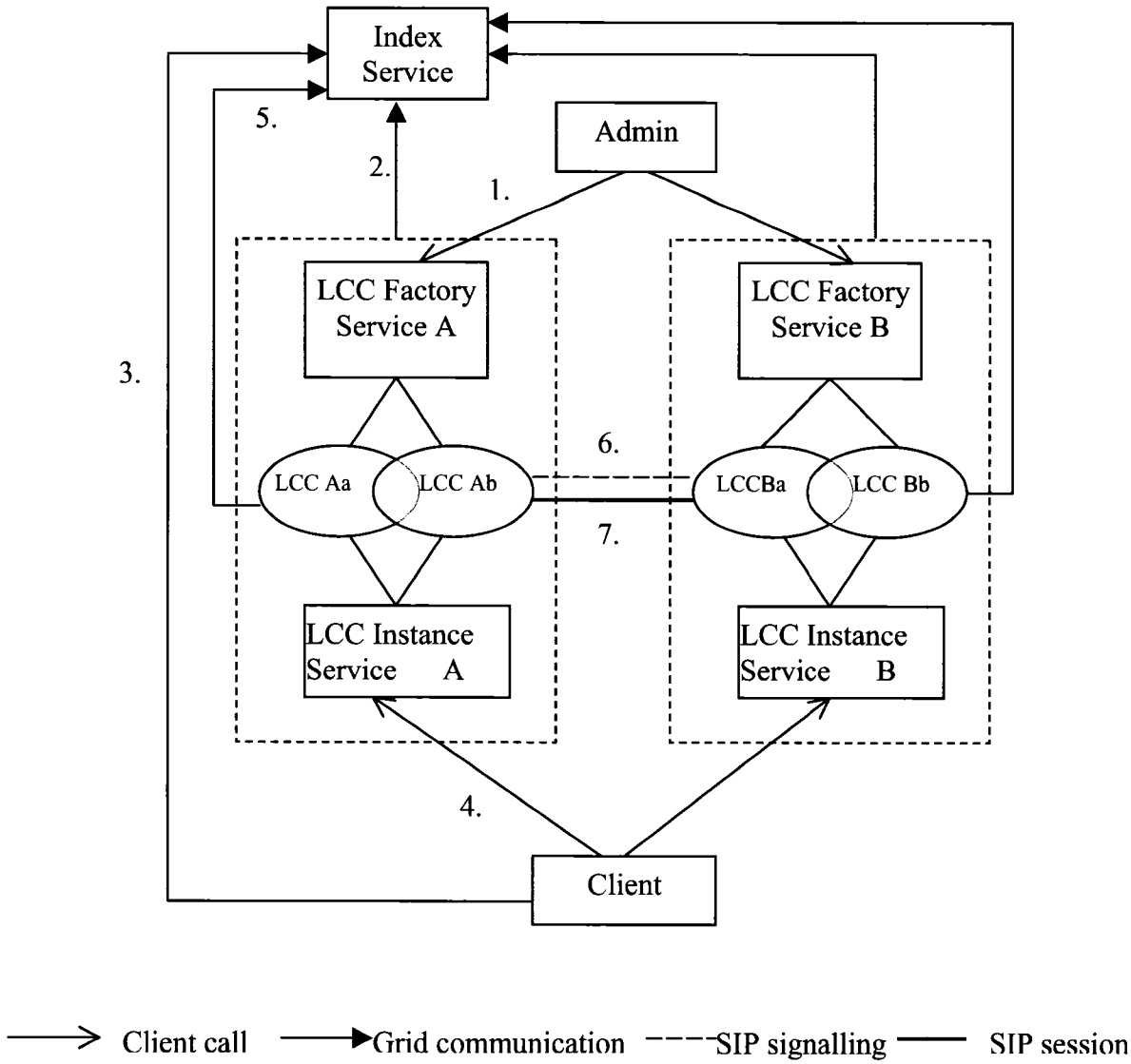

Figure 4-1. Architecture Overview of Grid –enabled CI

- LCC Factory Service: The LCC Factory Service creates LCC resource instances. The Grid administrator who manages the computing resources and services creates the LCC resource remotely or locally. Each LCC resource has its properties, most properties is kept from [6]. Moreover, four resource properties mentioned above are added and registered in Index Service during LCC's creation.
- LCC Instance Service: The LCC Instance Service provides Web services interface to outside and can accept from client application like creating a conference, adding an Attendee and making an announcement. The client can interact with LCC resource using LCC's endpoint reference gained from Index Service query.

These services are designed to provide primitive operations in their interface. The client only requires the WSDL definition to effectively exchange data with the service. The Web services interface provides its inherent interoperability and flexibility.

A typical conferencing can be carried out as follows (refer to Figure 1): the Grid administrator creates LCC resource using LCC Factory Service (step 1), during which LCC resources are registered in the Index Service (step 2). Then the client queries the Index Service to get the endpoint reference of one available LCC (step 3), and proceeds to instruct the LCC Service to create conferences using the endpoint reference (step 4). The LCC then queries the Index Service to get endpoint references and global security principals of other LCCs, which have the same conference subject (step 5), and signal all remote hosts derived from these LCCs' endpoint reference to create communication sessions using SIP (step 6). Once created signal channel with other peer LCC, the LCC create session channel using Generic Security Service (GSS)(step 7). Therefore, a fully distributed mesh conference model is created.

4.3 The Architecture of Grid-enabled SIP-RTI

In Grid-enabled SIP-RTI, The RTI component are built on top of the Grid-enable conferencing infrastructure and deployed in Grid environment. LRC provides a set of call back functions in Grid services interface, so LRC and LCC can interact using secure grid

services. The Grid-enable CI provides a common secure communication mechanism for interoperability of RTI.

The proposed architecture of the HLA components over Grid is illustrated in Figure 4-2. All components are deployed in Globus Toolkit 4(GT4)[4] container. Local RTI Component (LRC) was designed based on the WSRF implementation that use factory/instance pattern.

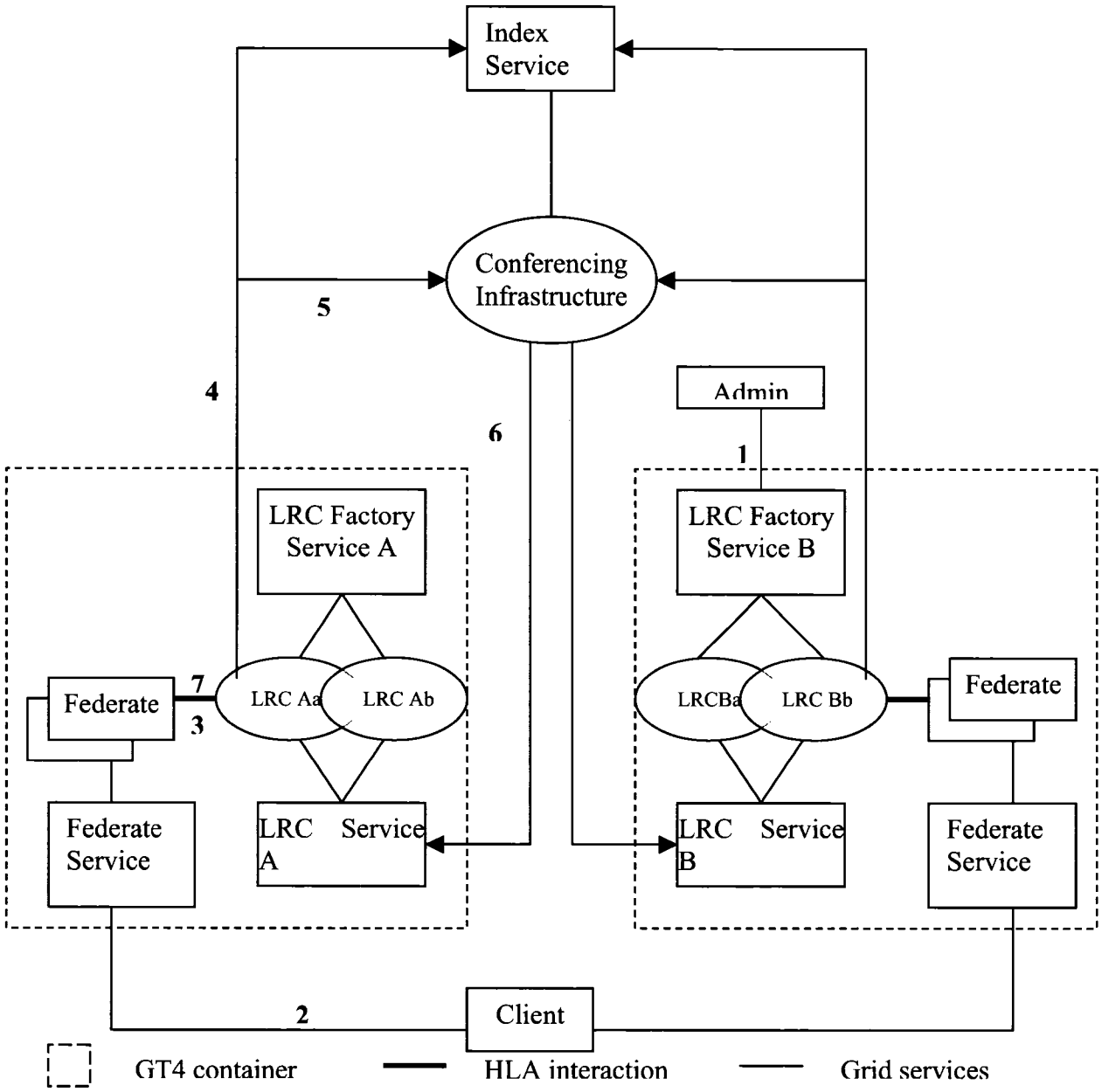

Figure 4-2. Architecture Overview of Grid-enabled SIP-RTI

Similar to LCC, the LRC component consists of several key Grid services, the LRC Factory Service and the LRC instance service, which are all implemented based on WSRF. The Index Service acts as an information server to facilitate dynamic discovery of the LCC resources in Conferencing infrastructure.

- LRC Factory Service: The LRC Factory Service creates LRC resource instances. The LRC resource is created remotely or locally by the Grid administrator who manages the computing resources and services. Each LRC resource implements the same RTI semantics according to the common underlying conference semantics. It interacts with federate and LCC.

- LRC Instance Service: The LRC Instance Service provides web services interface to conferencing infrastructure for response or request from other LRC.

- Federate Service: The only function of the Federate Service is to invoke federate. To invoke federate to start or join federation, Client have to invoke this Grid Service with valid credential.

Assuming conferencing infrastructure has been set up, a typical federation can be carried out as follows (refer to Figure 4-3): the Grid administrator creates LRC resource using LRC Factory Service (step 1), Then the client invokes Federate Services to create and initialize federate instance (step 2). Federate get the RTI ambassador from one available LRC. Then federate invokes RTI services on the LRC (step 3). LRC interprets the Federation Object Model (FOM) according the conference semantics, query available LCC from Index Service (step 4), and invoke one available LCC to participate conferences for communicate with other LRCs (step 5). Conferencing infrastructure works like a black box to LRCs. When get other LRCs request or response, LCC invoke LRC service to send messages (step6). LRC then notify corresponding federates through HLA specified services (step 7).

In this architecture, LRC and LCC are WS-Resources and can interact with each other through Grid Services. The Grid services provide high portability and flexibility to these components implementation.

4.4 Summary

This chapter presented the architecture overview of our Grid-enabled software conferencing system and Grid-enabled SIP-RTI. We started with the detailed analysis for our motivation to apply Grid technologies to software conferencing system and SIP-RTI that are presented in [5]. Then we described the architecture of conferencing system and SIP-RTI that were built on top of Grid. Also we briefly described the main components and the workflow in this architecture.

From the analysis that are presented in this chapter, we can find out that our work sit on top of the Grid middleware and under the simulation application layer. Figure 4-3 shows the layered framework for distributed simulation on Grid.

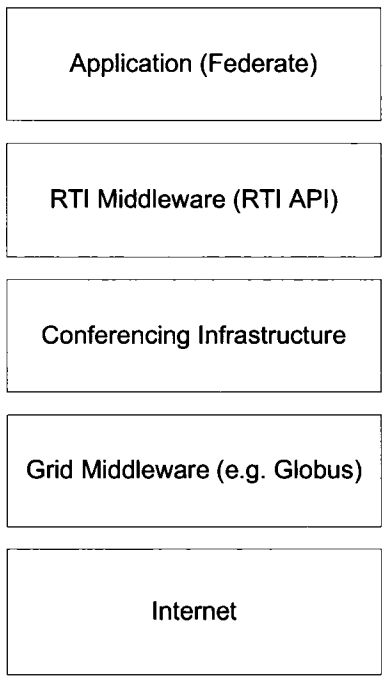

Figure 4-3 Layer framework for distributed simulation on Grid

Chapter 5 Design and Implementation

The Grid-enabled CI and SIP-RTI are extensions of SIP-RTI [5]. The extensions include three parts. First, to facilitate deployment management of LRC and LCC, secured Factory service and Instance service were developed for each of them. Also LCC and LRC are modified to be able to register its instances in Index Service and can be operated by their Factory service and Instance service. Second, to enforce security on the session channel between LCC instances, LCC are upgraded to implement secured session channel. Finally, the inter-LCC XML Data Type Definition is upgraded to support the extensions. This chapter discusses the design and implementation of the extensions. The major components that are implemented and how they interact with each other are also discussed.

5.1 Design and Implementation of Grid-enabled software Conferencing Infrastructure

The conferencing services provided by CI [5], the way to control call synchronization and concurrency in one LCC and to distribute data in LCCs to maintenance global state are kept. This grid-enabled CI is designed in service-oriented architecture to provide portability and optimum deployment management. At the same time, It enforces security to SIP session by make use of GSI.

There are three most important concerns for Grid-enabled CI. One is the creation, management and interface of LCCs. To make it portable, the LCC need to be created and deployed independently and have a platform-neutral, language-neutral interface. Section 5.1.1 discusses the major WSRF-compliant components for this concern. The second is the inter-LCCs message synchronization of the local LCCs in the same node. In grid-enable CI, multiple LCCs can be created as required. To synchronously maintain global state in each local LCC, the process of receiving messages need to be kept in order of FIFO to all local LCCs. Section 5.1.2 discusses the implementation for this concern. The third is the security concern for the secure inter-LCCs message communication on the SIP session channel. It will be discussed in section 5.1.3.

5.1.1 The major components

In this section, we will look at the major components in this service-oriented architecture.

5.1.1.1 Index Service

The Globus Toolkit 4 is an open source toolkit organized as a collection of loosely coupled components which fall into five broad domain areas: Security, Data Management, Execution Management, Information Services, and Common Runtime. The information Service in GT4, referred to as the Monitoring and Discovery System (MDS) , is a suite of web services to monitor and discover resources and services on Grids. The Index Service is one of the components that is used to collects information about grid resources and publishes that information as a service group. Client programs use resource property queries or subscription/notification to retrieve information from the index. Therefore, the default Index Service in GT4 satisfies our requirement of Index Service for application conferencing. Once one LCC is registered, there will be one Index entry created in the Index Service (Figure 5-1). The information kept in the Index entry includes the endpoint reference of the LCC and its registered resource properties, such as conference subject, principal, availability and identifier.

5.1.1.2 LCC Factory Service

The sole purpose of the LCC Factory Service is to create LCC resources that wrap and manage conference objects. It contains only one method that can be accessed by Grid administrator.

Upon accepting the request to create LCC resource, this Factory make a createLCCInstance call on LCCHome that hold references of all local LCC resources. After LCC is instanced and initialized, then this LCC is registered to Index Service.

```
<ns1:Entry>
    <ns1:ServiceGroupEntryEPR>
        .............
    </ns1:ServiceGroupEntryEPR>
    <ns1:MemberServiceEPR>
        <ns8:Address .....>https://..../LCCInstanceService</ns8:MemberServiceEPR>
        <ns9:ResourceProperties .....>
            <ns1:LCCResourceKey xmlns:ns1=....>xxxxxxxxx</ns1: LCCResourceKey>
        </ ns9:ResourceProperties>
    </ns1:ServiceGroupEntryEPR>
    <ns1:Content xsi:type="ns11:AggregateContent: xmlns:ns11="....">
            <ns11:AggregatorConfig>
                (registered resource properties)
            </ns11:AggregatorConfig>
            <ns11:AggregatorData>
                (the value of the registered resource properties)
            </ns11:AggregatorData>
    </ns1:Content>
</ns1:Entry>
```

Figure 5-1 Index entry in Index service

5.1.1.3 LCC Instance Service

The LCC Instance Service is the front door to provide conference control operations for all local LCCs. It is used to bridge the conferencing request to responsible LCCs, which in turn call their enclosing conference objects. When receive client's conference request which include endpoint reference of LCC, it first query the reference map of LCC resources to find the responsible LCC according the resource key enclosed in endpoint reference, then it maps the request to the destination LCC which in turn transfer the request to the destination conference object according to the conference ID. The Relationships between the Factory Service, the Instance Service, the Resource Home and the Resource is showed in Figure 5-2.

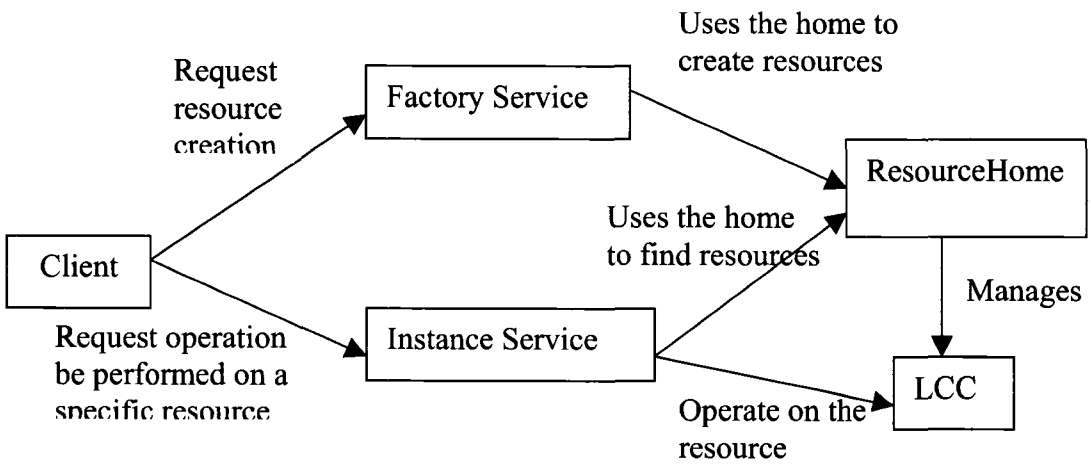

Figure 5-2 Relationships between the Factory Service, the Instance Service, the Resource Home and the Resource.

5.1.1.4 LCC ResourcesHome

The LCC ResourceHome do the real work to create LCC resources for Factory Service and find LCC resource for Instance Service. Besides manages the LCC references map, it also contains some node-wide conferencing helper objects, which applies to all local LCCs in one node. Figure 5-3 shows the high level view of LCC ResourceHome.

Because of the flexibility of the grid resources, there could be multiple LCCs in one grid node. The LCCs in one node can be regarded as LCC group .One LCC group share the same processors for the communication among LCCs. The processors include SIPHandler, MessageCentre and MessageReceiveCentre. The SIPHandler manages all SIP transactions for grid nodes. MessageCentre is used by all local LCCs to send messages to other LCCs. The MessageReceiveCentre implements a server, which accepts other sLCCs' request sent from MessageCentre to create security context between the LCC peers. After security context is established, conference message session channel is

established and supported by separate "MsgIOThread" thread objects that implement socket based input/output methods and are spun off by the MessageReceiveCentre in accepter side server and MessageCenter in requester side. The RunParserTDQ is also a separate thread. Along with the ParserToDoList queue it handles the synchronization of the messages received by MessageReceiveCentre. They must be processed in FIFO order to ensure proper state is maintained between all participating sLCCs.

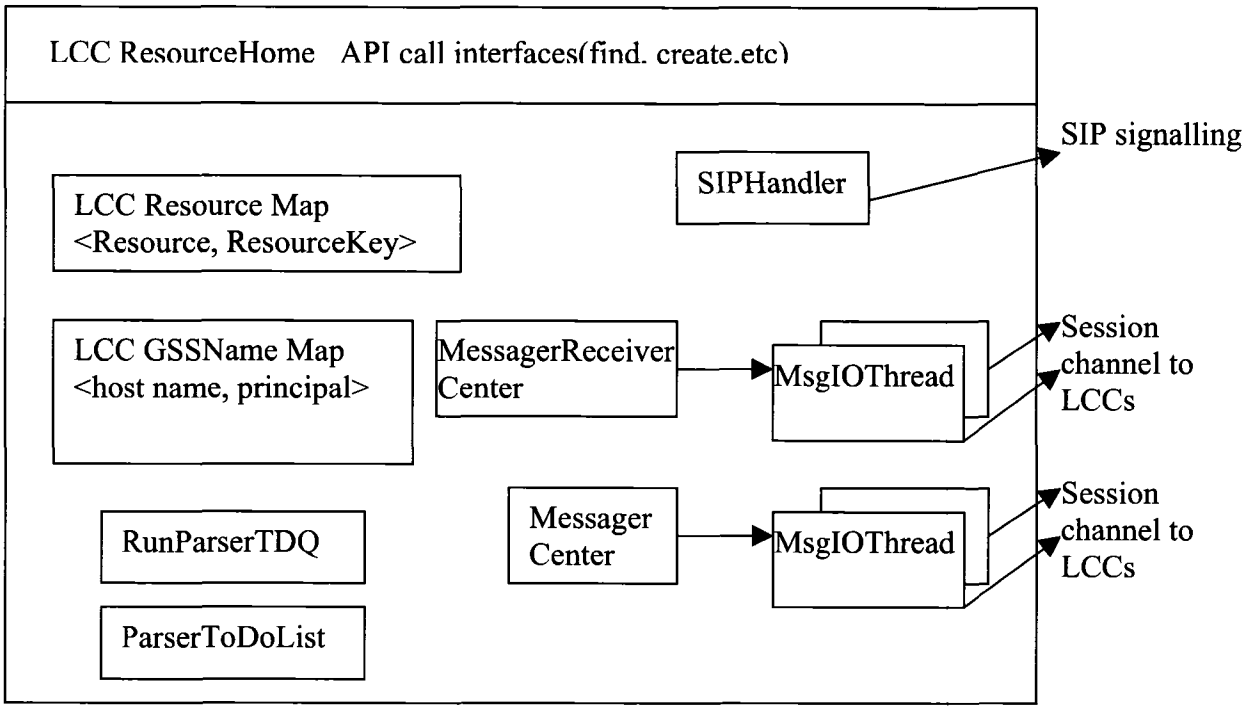

Figure 5-3 LCC ResourceHome structure

5.1.1.5 LCC Resource

The LCC resources provide the implementation to the LCC Instance Service interface. Figure 5-4 shows the high level view of LCC resource implementation.

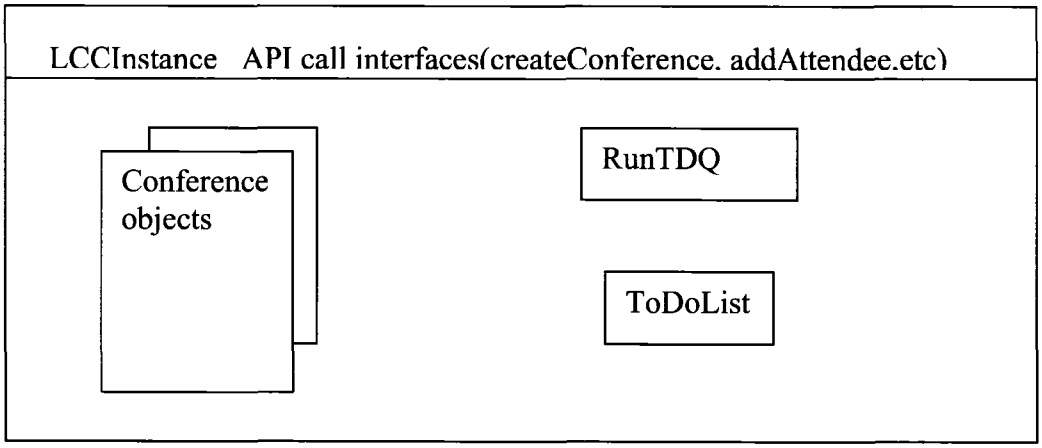

Figure 5-4 LCC Resource implementation

The RunTDQ is also a separate thread. Along with the ToDoList queue it handles the synchronization of some user calls on the single LCC API. They must be synchronized with responses from the other sLCCs to ensure proper state is maintained between all participating sLCCs. Conference objects handle all state changes, maintenance of local and global state information and other activity related to carrying out API calls. They, effectively, each implement a conference. All user conferences are subordinate to the ROOT conference, which is under the LCC's control [5].

5.1.2 Implementing Inter-LCC Message Channels

Before any messages can be sent, communication channel must be established among all participating LCCs. When a LCC joins a conference, other participating LCCs are either remote or local. For the remote LCCs, joining LCC use SIPHandler to setup a signaling channel with remote grid nodes and negotiate for creating messege channel. The setup of signaling channel is depiced in [5]. In this implementation, the message session channel is secured. Therefore, It is more complex to setup the secure message channel than that of

[5]. The security protection described in Section 5.1.4 discusses the setup of secure message channel. For the local LCCs, joining LCC directly start to setup the secure message channel.

5.1.3 Message Synchronization for Local LCCs

In this Grid-enabled CI, multiple LCCs can be created in one grid node as required. They are under the same node-wide processors control. They communicate by passing messages in SIP session. These messages contain important global states, hence affect the destination LCCs local state. We need to process the messages in order of FIFO to have each message processed in all destination LCCs before process next message. When MsgIOThread receive a message, it places the message to a Queue object, ParserToDoList. The thread of RunParserTDQ is responsible to route the messages in ParserToDoList in order.

The LCC use XML-based messages passing to communicate. The LCC can service multiple subjects, but one time only can service one. Meanwhile, although all local LCCs in one node are under control of the same processors, they may work for the different subjects. Therefore, the inter-LCC messages must contain subject (FederationExecution) name and LCC name. Figure 5-5 shows a sample of XML announcement message.

```
<?xml version="1.0"?>
<!DOCTYPE message SYSTEM "LCCMsg.dtd">
        <message>
            <fedName federationName="federationName"/>
            <lccTo  lccToName="lccToName"/>
            <lccFrom  lccFromName="lccFromName"/>
            <conference confID = "myConfNum"/>
            <reply yn = "reply"/>
            <announcements>
                <announcement the_announcement = "announcement" />
            </announcements>
        </message>
```

Figure 5-5 An Announcement Message

As shown in Figure 5-7, the subject name, called fedName, conference name and LCC name are included in message. If LCC name is not provided, then this message is a broadcast message to all local LCCs that have the same fedName. Otherwise, only the designated LCC receives and process the message. Figure 5-6 shows the workflow on the message processing.

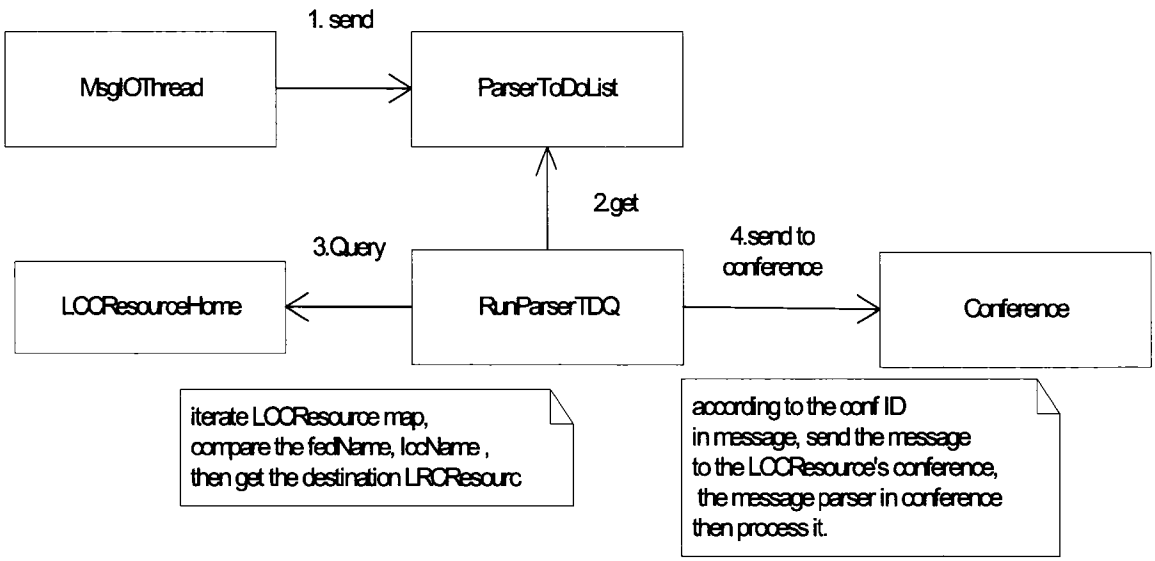

Figure 5-6 the Workflow of Received Message Processing

5.1.4 Security Protection

As a grid application, LCC resources are going to be accessed by a lot of different organizations. This poses a lot of security challenges. Grid Security Infrastructure (GSI) , the basis for GT4's security, allow us to overcome the security challenges, including authentication, integrity, privacy and credential delegation. GSI is implemented on top of the Generic Security Services application programming interface (GSS-API)[20]. GSS-API is both transport and security mechanism independent. Currently, GSI uses raw TCP sockets and the Nexus communication library [21] to move tokens between communication peers, uses the authentication protocols defined by the Secure Socket

Library (SSL) protocol [22] which is based on X.509 certificates for authentication. So GSI support Public Key Cryptography. Public-key systems can guarantee all of the three pillars of secure communications: authentication, privacy and integrity.

It is easy to add security to a grid service through configuring GSI. Moreover, GSI provide mutual authentication methods and protection level on a per-operation based. That is for all operations defined in grid service interface, their authentication methods and protection level can be different. It provides a flexible way to balance security level and communication efficiency.

When this conference system is used to communicate sensitive data, we should take three security policies into most consideration. The first is only the user who has valid certificate can query the Index Service to get the participating LCC list, and this communication should be secured. The second is only the user who has valid certificate can access LCC resource, and then conference. The third is the messages exchanging in SIP session should be secured.

The first two security policies are easily implemented by secure grid services through configuring GSI. So the most concern is about the SIP session. The Session Initiation Protocol (SIP) is a control protocol that supports the creation, control and termination of multipoint, multimedia communication sessions. It is a flexible protocol capable of supporting any application requiring an underlying communication infrastructure. There are a number of ways to provide conferencing with existing SIP mechanisms. Because of its growing popularity in the telecommunications sector, it is gaining increasingly widespread support, which has also resulted in the rapid development of extensions to it while maintaining the simplicity of its core specification.

However, due to the absence of a universal trust infrastructure, end-to-end authentication of communication partners is one of the fundamental problems for SIP. So we need a security mechanism to secure the real session stream to mitigate authentication problems for SIP. Because GSI integrate a Public Key Infrastructure (PKI), We explore the GSI

library and use GT4's implementation of GSS-API to implement secured communication in SIP session.

Using GSS-API to secure the communication is to establish security channel by creating security context that provides the security services, such as integrity, privacy and mutual authentication, etc. To create the security context, client has to provide four parameters: GSSManager, acceptor's GSSName, Oid, GSSCredential[jdk]. GSI provides all these interface implementation.

All LCCs in the same node have the same container credential, thus have the same GSSName. This GSSCredential can be retrived by from container's Certificate and private key file, or from credential proxy file. GSSName can be created using the principal in GSSCredential. However, the LCCs are distributed on grid nodes. The dynamic property of the LCCs determines they can be server or client. Therefore, the main issue is how to get their peers' GSSName or principal. In Figure 5-4, LCC ResourceHome contains a GSSName map, which holds the entry of host name of grid nodes and their principals. All LCCs in one node share the same principal and correspond to one map entry. To dymically locate LCC's GSSName, LCC's principal is set as one LCC resource property and is registered to Index Service. Figure 5-7 shows LCC's resource properties. The property of subject stands for the common interest of conferences held in the LCC (In case of SIP-RTI, Subject stands for federationExecution name); The property of isTaken tell us if this LCC has been used for conferencing; The property of dnName stands for the distinguish name contained in x.509 certificate and is used to create GSSName object.

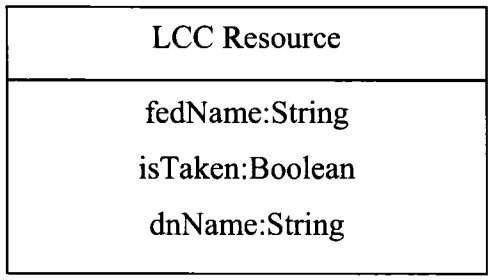

Figure 5-7 LCC Resource Properties Registered in Index Service

When one LCC want to join a conference, it first queries Index Service according its interest, which is LCC subject property. Then it gets all available LCCs that are not taken, with their dnName. Now it retrieve the host name of all the grid nodes from the LCCs' address and put the pair of host name and dnName to the GSSName map in LCC ResourceHome. Then it uses SIPHandler to start signaling other LCCs' grid nodes for session creation negotiation. If negotiation is successful, this LCC start to create session using MesssageCentr by creating security context based on its peers' dnName. The acceptor node's MessageReceiveCentre also create security context. This two-side security context provides all security service for the session. Figure 5-8 shows the workflow for this process.

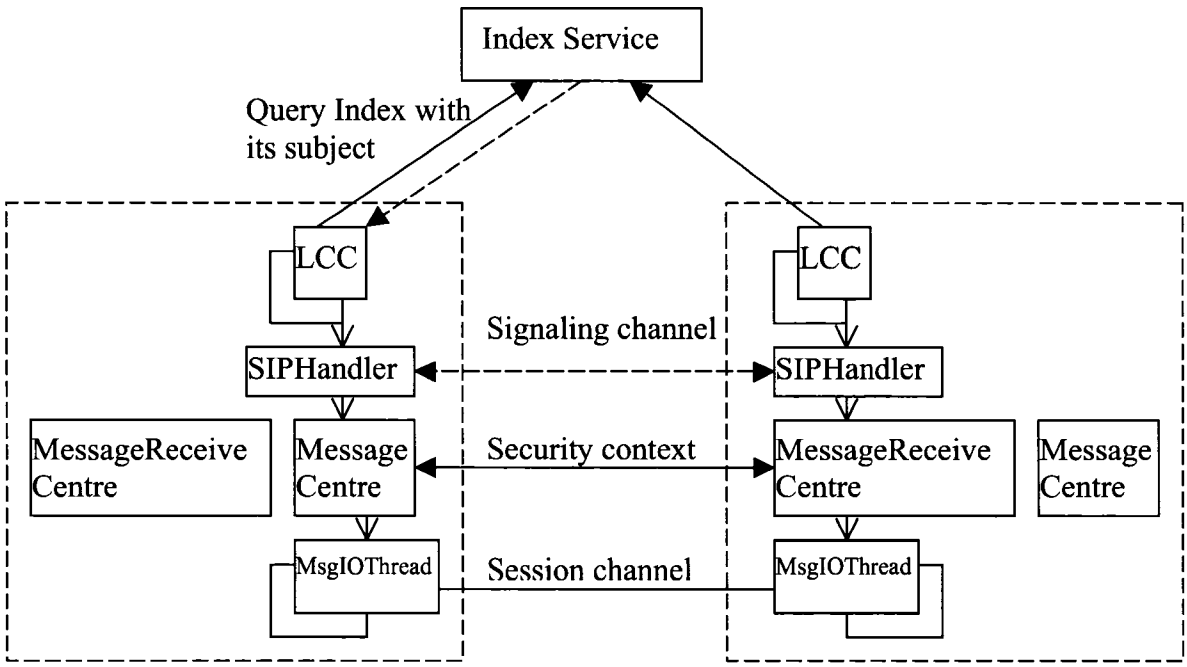

Figure 5-8 Secure Workflow When One LCC Is Joining A Conference

5.2 Design and implementation of Grid-enabled SIP-RTI

In Grid-enabled RTI-SIP, the LRC acts as a surrogate to transfer the HLA services invocation from federate to conferencing service invocation in the underlying distributed LCC, which in turn take over the federates interaction control in CI. Grid administrator can independently and dynamically create the LRC in any grid node. Multiple LRCs can be created in any grid node as required. To provide user-level security to federate, the federate is also deployed in grid and locates in the same process as LRC. Multiple federates can invoke one LRC's HLA services, and multiple LRC can use one LCC during a federation execution. Figure 5-9 shows the cardinality relationship between federate, LRC and LCC.

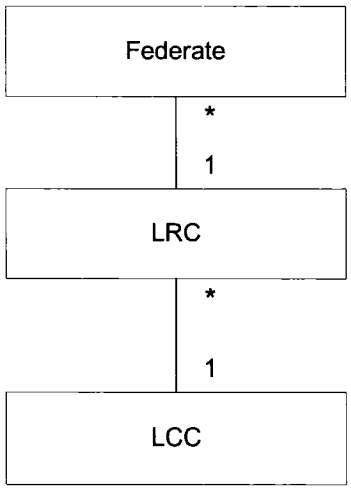

Figure 5-9 the Cardinality Relationship Between Federate, LRC and LCC

In federation execution, the federate choose an available LRC and make RTI services call on it, which in turn choose an available LCC and make conferencing service call on this LCC. In this way, the FedExec process move down to underlying distributed CI. CI provides secure grid services interface for user application, and secure communication within CI.

There are three main concerns in this Grid-enabled SIP-RTI. One is how to deploy federate in Grid and how federate choose LRC. Section 5.2.1 discusses the implementation of this concern. The second is how LRC locate an available LCC and provide a portable interface for LCC's callback functions to secure the callback. The implementation of this concern will be discussed in section 5.2.2. finally, section 3.2.3 describe the detail of the security solution for SIP-RTI by analyzing the possible threat during federation execution.

5.2.1 Deploying Federate in Grid Environment

Grid provides a common open standard security infrastructure crossing multiple security domains. By deploying federates in grid Environment, different user will have different security level according to their own grid-wide identity. It benefits the federate's reusability in wide simulation communities. The implementation of federation is independent with RTI and is not touched. As shown in Figure 4-1, FederateService only provide a grid service interface to federate user. Its implementation simply loads the federate's main class and invoke its main method to instantiate one federate instance. During the federate instance creation, this federate instance looks up the LRCResource Map in LRCResourceHome and gets an RTIambassador of an available LRC from the possible multiple LRCs in the same process. Figure 5-10 shows this process to get RTIambassdor.

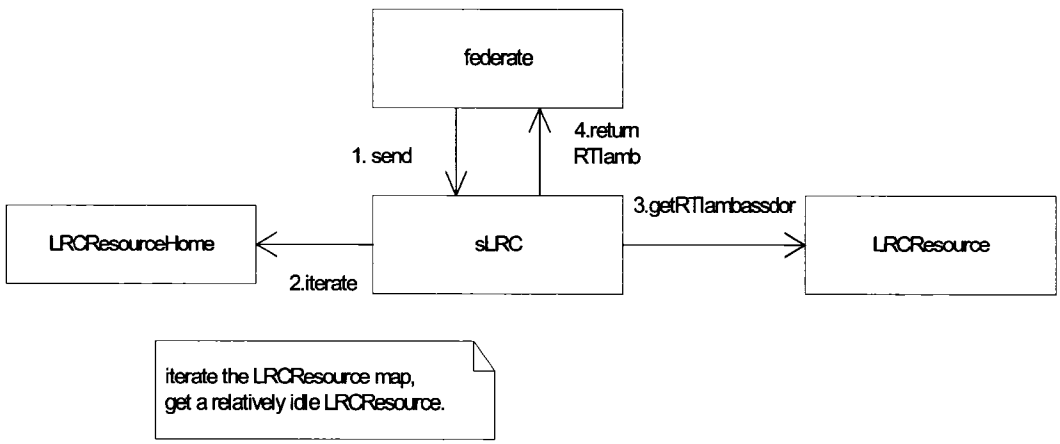

Figure 5-10 the Workflow of Getting RTIambassor

5.2.2 the major components of Local RTI component

Similar to CI, to create and manage the LRC, two key Grid services interface are designed and implemented, namely the LRC (Local RTI Component) Factory Service and the LRC instance service, which are all WSRF compliant interfaces.

LRCResourceHome are the real implementation responsible for create and manage LCCResources. Figure 5-11 shows the high level view of local RTI component. The Outer line represents the Java package that contains all sLRC classes.

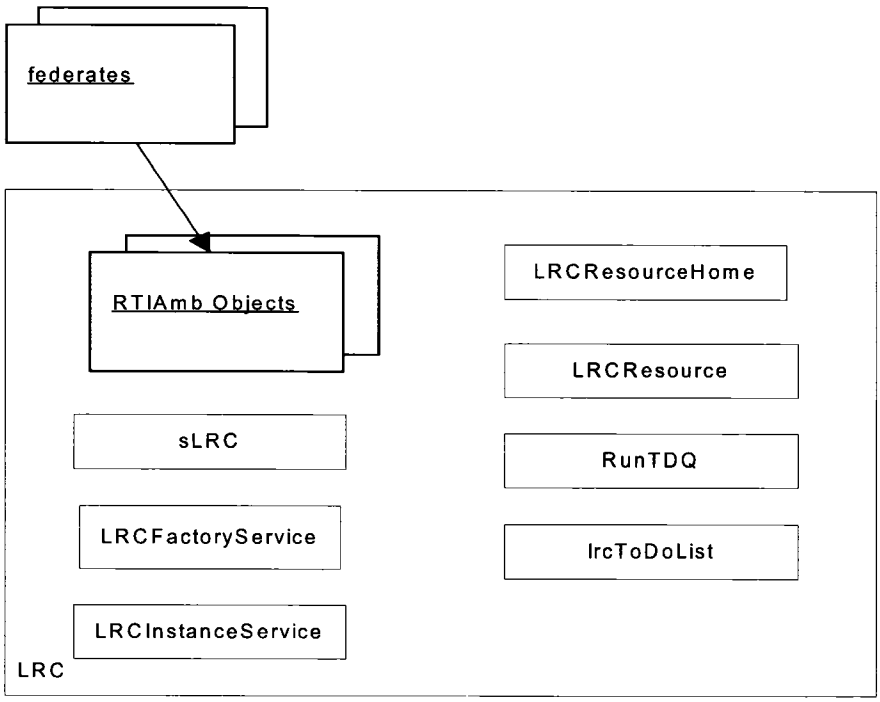

Figure 5-11 the High Level View of LRC Package

Except two grid services, LRCResourceHome and LRCResource, which are added, sLRC is totally changed so that it was kept only to keep the integration of federate and LRC untouched. Other classes are kept untouched. The detail of their implementations and functions is described in [5]. Figure 5-12 shows the structure of LRCResourceHome

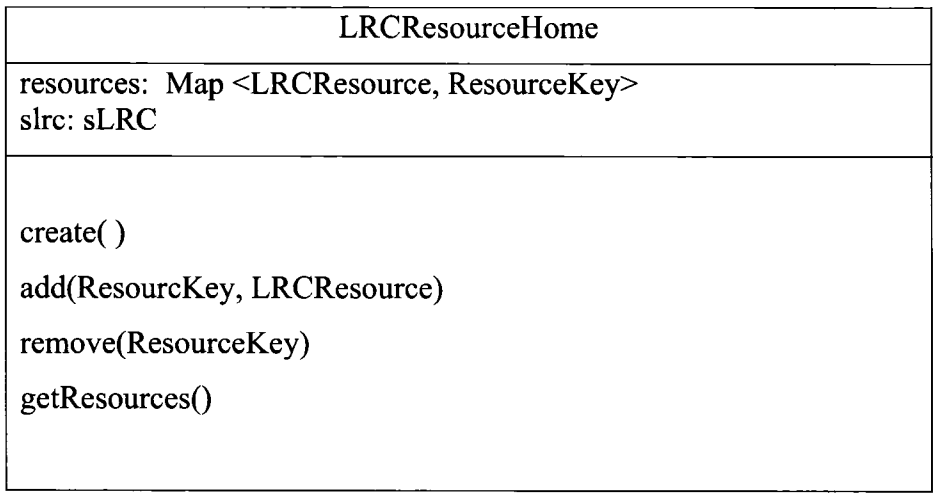

Figure 5-12 Structure of LRCResourceHome

The same as the implementation of in [5], LRC maintains the designators mapping (Figure 3-3) to transfer the HLA services call from federates to conferencing services call in LCC. The creation and interfaces of LRC can be referred to LCC in last section. Once federate make the createFederationExecution call on an idle LRC, the LRC firstly query the Index Service to discover an available LCC according to the registered resource property in Index entry, then make createConference API call to create conference map in the LCC according to the FOM and get the handle of conferences and federates. If the LRC is serving other federates, when a federate make createFederationExecution API call on it for the same federation, the LRC only generate ConfID for the object classes and their attribute, but not make createConference API call on its LCC. Figure 5-13 shows the workflow that LRC get an LCC when its createFederationExecution is called at first time by federates.

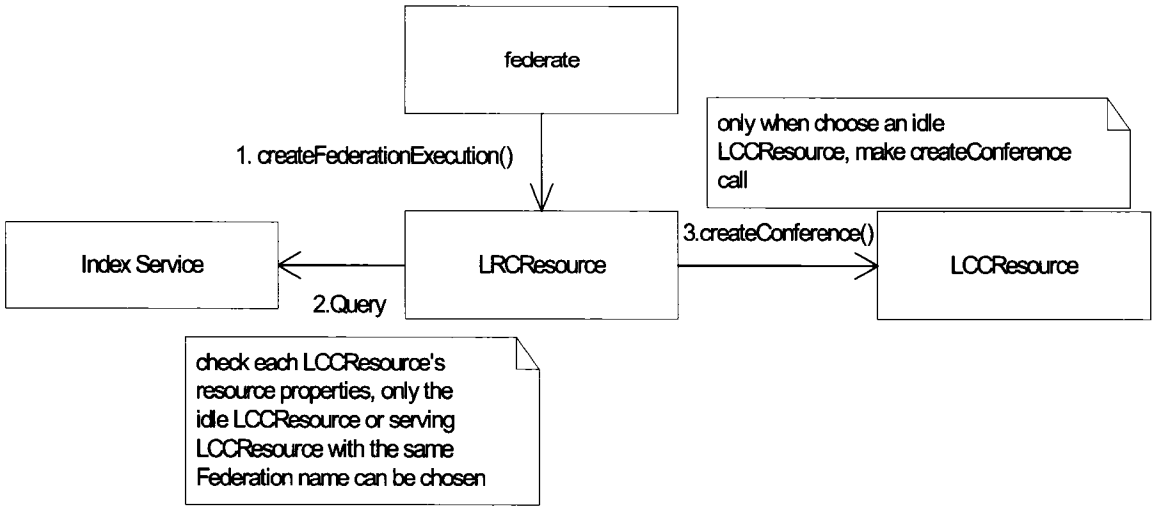

Figure 5-13 the Workflow that LRC Get An LCC

5.2.3 Security Protection

GSI provides a common open standard security infrastructure. The common security layer will provide flexible solution to all security objectives.

Current security solution applies to the simulations whose users are mutually trusted. In this kind of simulation, the only security objective is to prevent data transferring in federation from leaking to other parties. We describe how LCC's communication is secured through analyzing the threats to the security of this Grid-enabled SIP-RTI. Figure5-13 shows the overall work flow among a federation. From figure5-14, we distinguish four kinds of interactions between components.

- Internal interactions within a process: The federate code calls LRC for HLA services requests , and LRC calls back to federate when messages from LRC are received.
- Interactions between LRC and Index Service: the LRC queries Index Service to locate an available LCC.
- Global interactions between processes: the LRC interact with LCC that locates in conferencing infrastructure.

- Interactions between LCCs: the distributed conference components communicate with each other through a session channel created by SIP signaling channel.

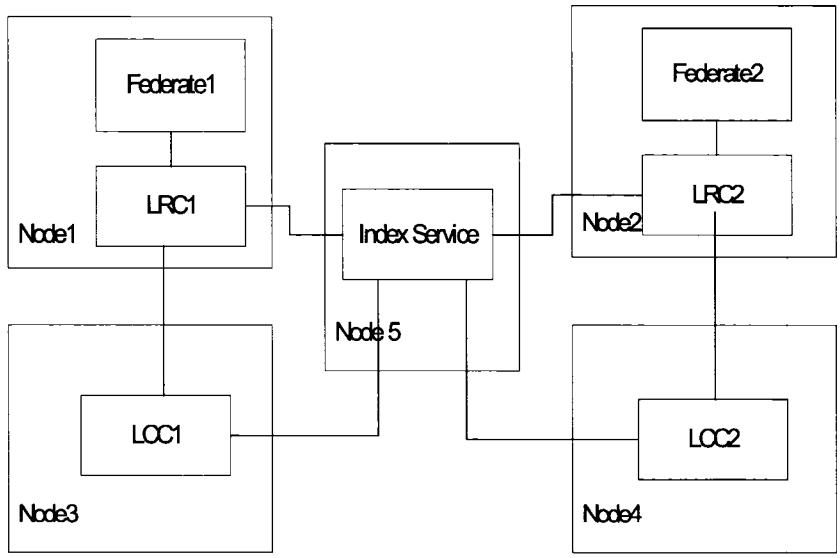

Figure 5-14 RTI work flow

Obviously, the last three inter-process interactions are subject to security threats. However, both the interaction between LRC and Index Service, and the interaction between LRC and LCC use secure grid services. Hence these interactions are secured. Meanwhile, The interactions between LCCs are also secured as discussed in section 5.1.3. Therefore, the security objective in mutually trusted simulation users are achieved. Because that deploying federate in grid environment provides the user-level security for simulation reusability, in fact, current implementation achieved the first level security objective and part of the fourth security objective.

5.3 Summary

This chapter presented the design and implementation of Grid-enabled software conferencing system and SIP-RTI in detail.

For the CI, we started with the detailed description for their main components, and then analyzed the implementation of the key workflow, such as the creation of message channel and message synchronization. Finally, we described why and how deploying CI on top of Grid achieves the secured message session channels. Since RTIs' communications were moved down to the CI, the secured session channels can satisfy the security need for RTI.

For the Grid-enabled SIP-RTI, we also started with the description of its main components, and then analyzed the possible security issues when this SIP-RTI was used in the simulations whose users are mutually trusted. In this case, only two interactions need security protection. One is the interaction between LRC and LCC; one is the interaction among LCCs. As discussed in CI, the communication between LCCs is secured by secured session channels. Meanwhile, the interaction between LRC and LCC uses Grid services, which provides strong security protection. Therefore, the RTI's security issue is resolved.

From the analysis presented in this chapter, we can find out that LCC can be discovered and used on demand, and the interoperability of RTI was resolved by making use of LCCs. Further more, the security issue of RTI was also resolved by take advantage of Grid services and GSI.

Chapter 6 Evaluation

In this section, we will present an example to demonstrate our Grid-enabled conferencing infrastructure are portable system that can be deployed independently to provide the interoperability for local and remote software (in this case, it is LRC). At the same time, it provide strong security protection to all communications, one of which is between LRC and LCC, the other is between LCCs.

In this example, a demonstration federation, consisting of three federates and simulating the sharing of one common object, was employed to display, and further test, the HLA object management services implemented. To demonstrate our Grid-enabled SIP-RTI is a generic solution for reusable simulation, we take an off-shelf federate that is developed independently according to the HLA interface specification by other people into this federation.

After describing the development and test environment, the following sections describe the testing done and the demonstration federation.

6.1 Development and Test Environment

A two-node test and development network was used (figure 6-1). All implementations were done in JAVA using various releases of the Eclipse 3 workbench [39]. Since our work is based on the implementation of [5], The SIP services implementation was kept using the JAIN SIP API v1.1 RI, which is a SIP API, based on the NIST SIP V1.2 parser and stack [40]. However, the session creation request is started from SIP negotiation channel initiator, not from SIP negotiation channel acceptor as [5]. Because only the SIP initiator has the GSS name of the acceptor, which was retrieved from Index and is required to initiate the security context between them. Development and test computers were Inter X86 Processor desktops, running Sun Solaris 10. A 100BASE-T (Fast Ethernet), LAN was used. Both machine are installed and configured the Grid middleware, Globus Toolkit 4. And install the SimpleCA in one of the machine, which is

the CA to sign the host certificate and user certificate. Please note that in order to use the services provided by GT4, a CA signed certificate is need to be in the host. Similarly, in order to use the Grid environment, a grid user need to have a CA signed user certificate. In this example, we setup the host certificate and user certificate for each host and each user. After the installation and configuration of GT4, we deployed the LCC and LRC module into GT4 container. Finally, we create three LRCResouces on each of the node, and three LCCResources in two nodes. The Index Service will hold information entries for each LCCResource. All the following testing assumes the above steps are accomplished. To demonstrate the flexibility and portability of our Grid-enabled software conferencing system, we use three kinds of configuration of LCCs.

- All LCCs are created on the same process as requesting LRC
- All LCCs are created on the remote process to requesting LRC
- LCCs are created on both the same process as requesting LRC and the remote process

Note figure 6-1 shows the third configuration of each node after the LCC, LRC and demonstration federation code was deployed and created on each.

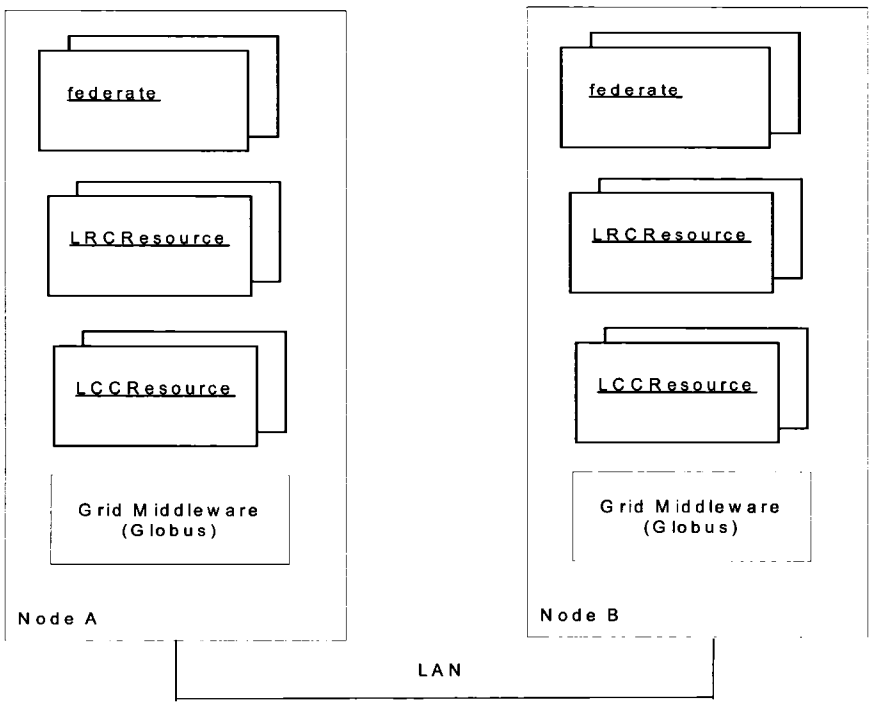

Figure 6-1 Development and Test Bed

6.2 LCC Testing by general client application

We developed general client application to test the Grid-enabled conferencing system. The LCC Testing was progressive. Each API service was added and tested. Conference creation was the first call implemented. It was used to create a 5-conference conference map in all LCC Resources. Then federation joining was tested by connecting the LCC Resources. The log shows that during join the federation, LCC Resource first retrieve all LCC resources with the same federation name by querying Index Service. Then create secured session channels to each node that have LCC Resources having the same federation name through SSL. If the node is remote, the SIP is used for negotiation to start the creation of the session.

After federation creation, the object management related services were tested. One client application adds speaker and floor in one conference located in one LCC Resource, two client applications add listener in the same conference located in other LCC Resources. The log shows all the expected global state transfer. Finally, the client application having speaker in the conference make an announcement in the conference. In consequence, the other two applications receive the announcement.

6.3 LRC Testing carried out in the Demo Federation

Testing of the LRC was carried out in the demo federation. Similar to that of the LCC, All object management calls were tested and verified. During LRC testing, LCC state was monitored by a monitor application in addition to LRC activity to verify correct behaviour in this scenario. All tests were run on LRCs and LCCs located on both two nodes. The output log shows that the federate first retrieve one idle LRC Reource and get its RTIAmb, and then using this RTIAmb to make HLA specified interface call on the LRCResource. This LRCResource proceed to retrieve one LCCReource (the idle is prior) and make corresponding Grid services invocation on the LCCResource. The final test

was the development and running of the demonstration test program which is now described.

6.4 The Demo Federation

To demonstrate our Grid-enable conferencing infrastructure can provide portable solution for HLA RTIs' interoperability and security, we set up one demo federation and test the object management related services. Simulation of a simple chatting scenario provides a suitable real-world situation. In this scenario an speaker federate update the state of one shared object at intervals, and the listener federates continuously receive this updated state through the LRCs and LCCs.

There is only one object class in the simulation, the STOP_WATCH object. The FOM used in this demonstration is contained in Appendix B. It can be represented using the HLA object class hierarchy diagram shown in figure 6-2. The resulting CI conference map is provided in Appendix C.

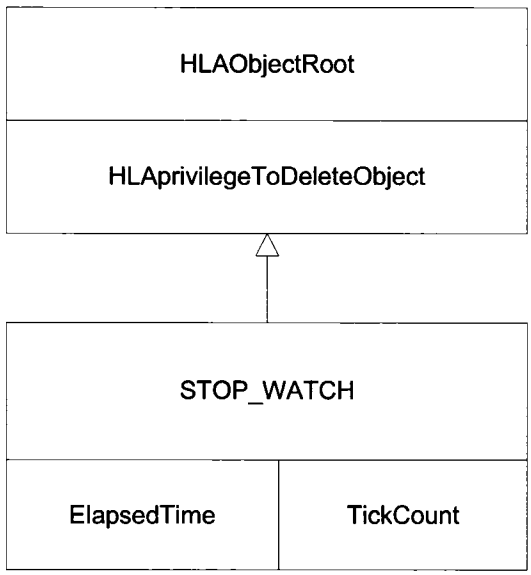

Figure 6-2 HLA Object Class Hierarchy Diagram of the Simple Chat Federation

The STOP_WATCH object has two attributes (in addition to the inherited HLAprivilegeToDeleteObject attribute):"ElapsedTime" represents the interval between two object state updates; "TickCount" represents the times of the updates. Both attributes are implemented as Java Integer class objects by federate.

The simple chat federation controls the synchronization of shared object state. It publishes ElapsedTime and TickCount by speaker federate, and then subscribe ElapsedTime and TickCount by listener federates. When start chatting, the speaker federate adds floor in the conferences of ElapsedTime and TickCount located in LCCs. Then start to update the state of ElapsedTime and TickCount. The listener federates will receive all the updates states.

The demo was successfully run with all federates joined before the simulation began running and with late joining listener federate. (The speaker federate cannot join "late" as it drives the simulation). The scenario demonstrates all aspects of object management related services and all possible types of transfer including transfer of participant list and floor list between federates. The output logs also show all the communications are protected by SSL.

It is considered a complete test of the LRC and LCC functionality provided and, as well, a successful proof of the application of a conferencing paradigm to distributed communication. It undoubtly demonstrate that our Grid-enabled SOA-based software conferencing system is a total portable solution to address RTIs' interoperability and security.

6.5 Demonstrating Interoperability between Different Applications

A monitoring application was developed as a simple non-RTI application to demonstrate the ability of the CI to support interoperability between different types of applications. It demonstrates this by communicating through a common CI with the SIP-RTI that

supports the simple chat demonstration. Figure 6-3 shows the configuration of the two demonstration applications.

The Grid-enabled SIP-RTI is implemented by the Grid-enabled RTI/CI combination and supports federate implementing simple chat simulation. Note that the monitor and the RTI middleware connect directly to the CI. The monitor program attends one conference in the CI. As a result, the monitor application receives all updates in the form of an inter-LRC Update message.

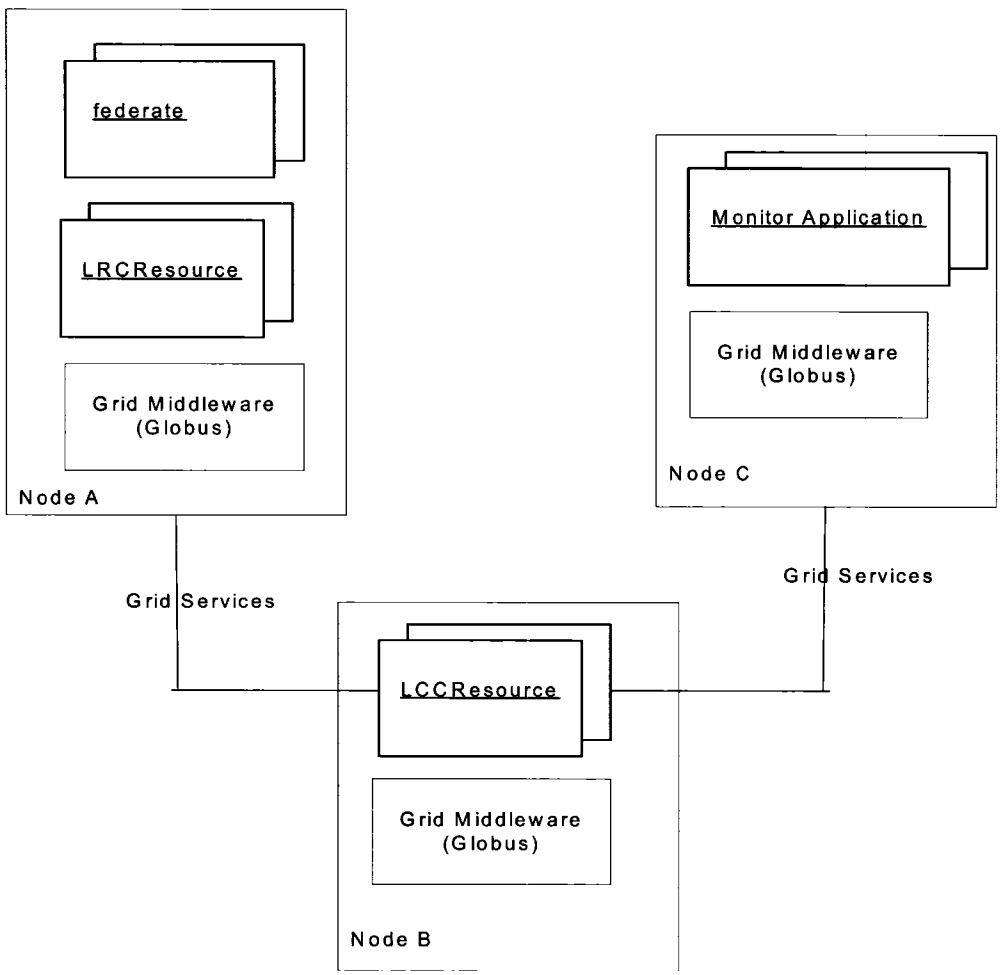

Figure 6-3 Demonstrating Interoperability

Further more, because our conferencing system provides grid services interface, The applications implemented by different languages also can interact this CI. Therefore, this Grid-enabled software conferencing system provides better interoperability than [5].

6.6 Testing Limitation

To evaluate our work in large-scale distributed simulation, we need set up the testing in different security domains. That means at least federates and LRCs should be deployed Grid environment that are installed on machines in different domains. This kind future testing would be more demonstration for the function of our work.

6.7 Summary

This chapter presented a simple chat demonstration federation to evaluate our Grid-enabled software conferencing system and SIP-RTI. The result shows that by deploying CI and RTI on top of Grid, all the three HLA issues, including RTIs' interoperability, RTI security and discovery of federation, are possible to be resolved for large-scale distributed simulation. Meanwhile, the result demonstrates that our Grid-enabled CI are a generic and portable system that can provide interoperability and secured interaction to general Heterogeneous applications.

Chapter 7 Conclusion and Future Work

HLA posses advanced mechanisms supporting distributed simulations, so execution of HLA based applications on the Grid should be natural extension of its usage. This research believe grid technology will be a good complement to HLA, thus refract and extend the SIP-RTI implementation model into grid. As a result, a portable Grid-enabled, SIP-based conferencing system was designed and implemented. In this Grid-enabled SIP-RTI model, RTI and Federate component were both deployed in the same process in grid environment. The LCC in CI can be deployed within any process in grid. The federate call the RTI services to LRC, which in turn call secured conferencing services to LCC in CI. The federation execution takes place in the CI by passing conference message in secure SIP session among LCCs. In the rest of this chapter, we will conclude this thesis with a brief summary, and then present any future work or ideas not implemented.

7.1 Conclusion

This thesis describes a portable grid-enabled SIP-based conferencing infrastructure, and grid-enabled SIP-RTI, which extends the implementation of SIP-RTI [5] for security and deployment management. Through the demonstration federation, we can conclude this thesis as following:

1. This thesis achieved part of security objectives in HLA simulation. When the users of federates are mutual trusted, all data transferred through federation can be secured. Meanwhile, the users can provide their privileges to federate. Therefore, federates can be reused at different security levels according to the users' credentials.
2. This thesis provide a flexible deployment management solution to SIP-RTI components: LCC and LRC. These components are decoupled by grid service interface. Moreover, the factory design pattern was used in their implementation, so their instances can be created in any node on demand.
3. This Grid-enabled SIP-RTI totally complies with IEEE standard. The interface with federate and LRC are kept untouched.

4. This Grid-enabled SIP-RTI can support off-shelf federate. In the demonstration federation, the federate was developed by third-party.
5. This Grid-enabled software conferencing infrastructure is able to provide secured interoperation among general applications. In the demonstration federation, LRCs and the monitor application can interoperate securely through the CI.

7.2 Future Work

As discussed in Chapter 3, large scale distributed simulation poses a lot of security challenges. Currently, this implementation solved the security issue in the simulations that are supposed to be mutual trusted and provides user-level security for simulation reuse.

However, Our Grid-based architecture is promising to provide solution to all the other security objectives. Referring to [21], We will present a list of unimplemented ideas to satisfy other security objectives discussed in Chapter3.

- **To Achieve the Second level Security**

GSI is based on PKI. So CAs can be arranged hierarchically to separate mutually suspicious organizations. The unclassified CA is at the root level, the more secret CA is at one lower level, and likewise the most secret CA is at the lowest level. The result is the members in the lower levels can access the resources in upper levels, but the members in the upper levels cannot access the resources in lower levels. So that the private information was kept in mutual trusted members determined by the CAs in the same level or lower levels. Figure 7-1 illustrates the idea to achieve the second level security.

In Figure 7-1, CA1 entrust CA2, CA2 is in lower level of CA1. So the resources that are authorized by CA2 form a enclosed VO, which is enclosed by a enclosing VO formed by the resources that are authorized by CA1. In this scenario, the resources authorized by CA2 can access the resources authorized by CA1, but the resources only authorized by CA1 cannot access the resources authorized by CA2.

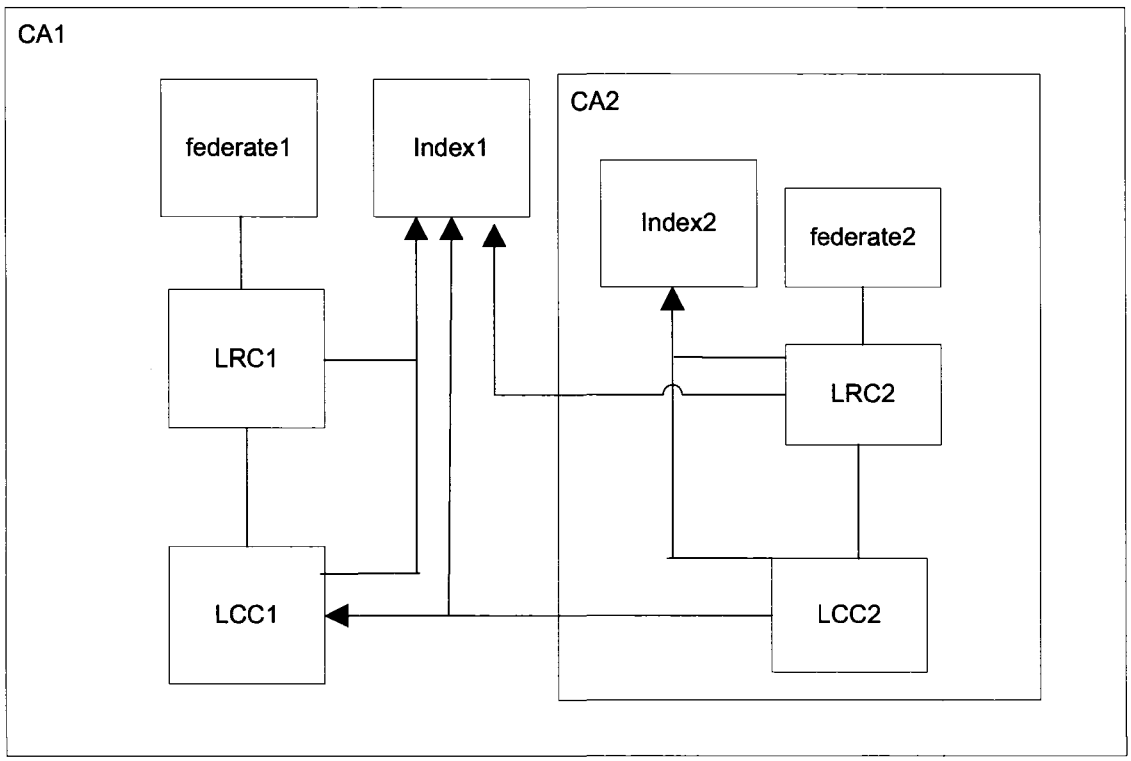

Figure 7-1 to Achieve Second Level Security

- **To Achieve the Third Level Security**

To fully support multi-level secure principles, security extensions to RTI and federate object models are required to allow federates with multiple security levels of data to provide the data to other objects [21]. First we could explore GSI library to provide a mechanism to request credential for the federate object model (FOM), then we need a mechanism like MyProxy [4] to store and retrieve the requested credentials on demand. Based on the data security requirements, the objects defined in FOM would need to request credential from the appropriate CA. Additionally the FOM would need to support

Access Control Lists (ACLs) that enable the federate and RTI to allow or deny access based on the provided authentication information, provided via the private/public key implementation. Also, LRC have to be able to retrieve the credentials for objects/attributes in FOM and delegate these credentials to LCC. Figure 7-2 shows the idea to achieve the third level security.

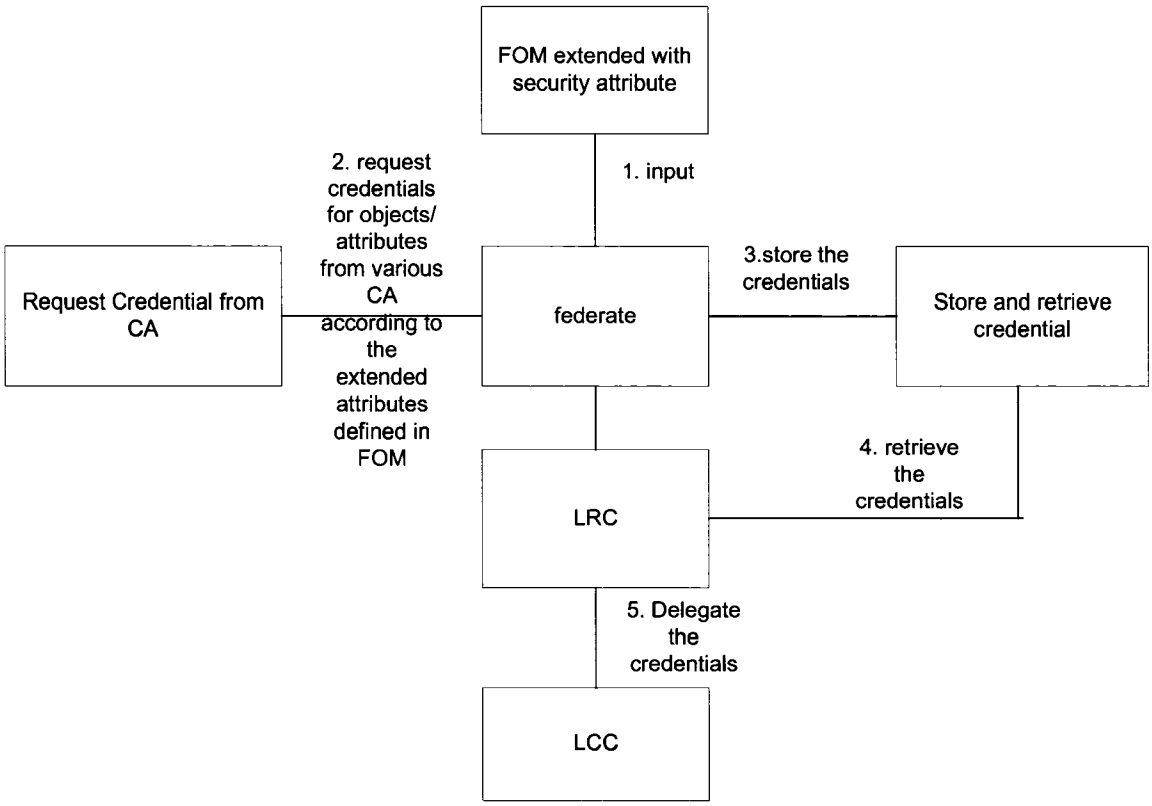

Figure 7-2 to Achieve Third Level Security

- **To Achieve the Fourth Level Security**

[21] can not achieve this objective due to IPSec limitation. However, Grid environment can make it. To reuse simulations at different security levels, the simulation user's authentication is required. In grid-based architecture, federate service provides the authentication function for simulations. Then the FOM would need to provide a mechanism through exploring GSI library for requesting its credential according to the authenticated user identity. So FOM need to be extended with attributes about user name

or group, by which federate can decide the credentials of objects/attributes based on the authenticated grid user. Figure 7-3 shows the idea to achieve the fourth level security.

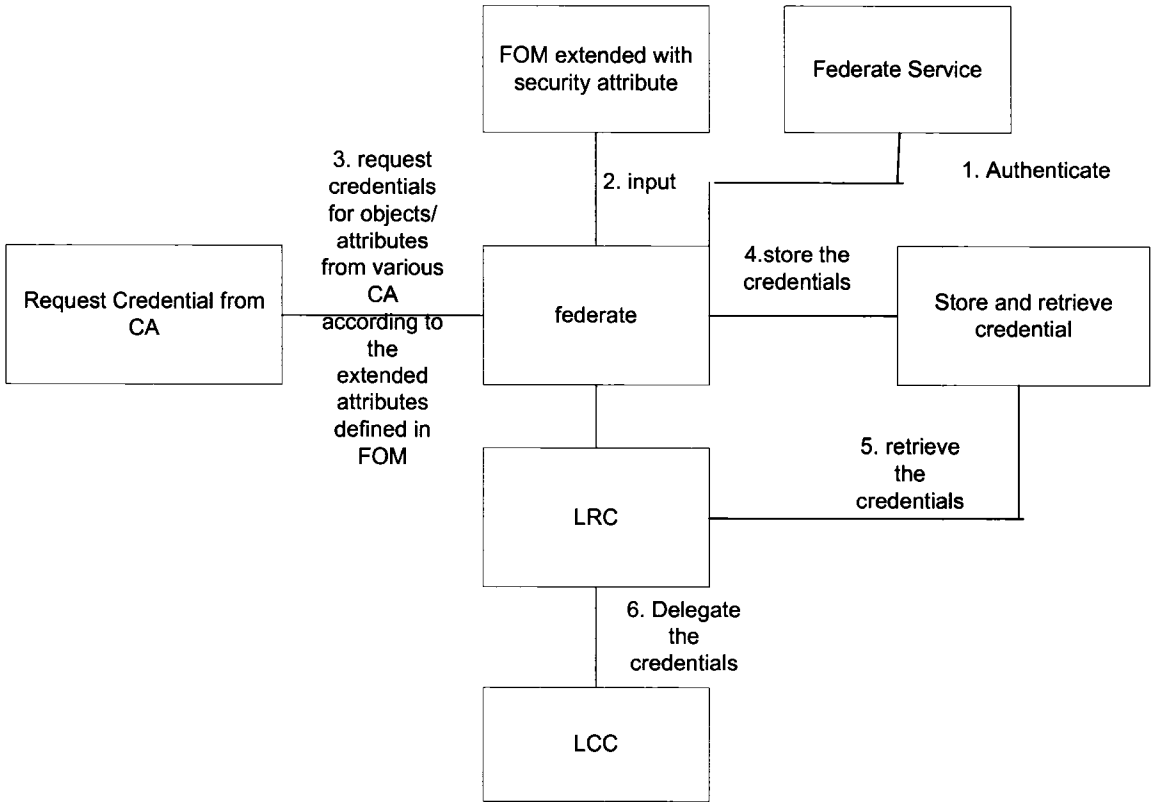

Figure 7-3 to Achieve Third Level Security

References:

[1] Judith S. Dahmann, Richard M. Fujimoto, Richard M. Weatherly, "The Department of Defense Hight Level Architecture", Proceedings of the 29th Conference on Winter simulation WSC '97, December 1997

[2] T. Pearce and N. Farid, "If RTI' s Have a Standard API, Why Don' t They Interoperate?, " in Proceedings of the Simulation Interoperability Standards Organization Fall Simulation Interoperability Workshop, 2004.

[3] Global Grid Forum(GGF). Http://www.ggf.org

[4] The Globus Alliance, [Online homepage] , Available at http://www.globus.org/

[5] Claude O. H. Van Ham, "SIP-RTI: A High Level Architecture, Runtime Infrastructure built on a SIP-enabled Conferencing Mechanism", Master's Thesis, Carleton University, 2006

[6] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of Supercomputer Applications, volume 15(3), 2001.

[7] Web Services. http://www.w3.org/2002/ws/

[8] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 2002. http://www.globus.org/alliance/publications/papers.php.

[9] OASIS, "WS-Resource specification". 1.2 Working Draft 03. Web Services Resource Framework (WSRF) TC. OASIS. March 8, 2005

[10] W3C, "Web Services Addressing (WS-Addressing)", W3C Member Submission 10, August 2004 , http://www.w3.org/Submission/ws-addressing/

[11] The Institute of Electrical and Electronics Engineers Inc., IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification, IEEE Std 1516.2-2000, New York, 2000.

[12] The Institute of Electrical and Electronics Engineers, Inc., IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, IEEE Std 1516-2000, New York, 2000.

[13] The Institute of Electrical and Electronics Engineers, Inc., IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification, IEEE Std 1516.1-2000, New York, 2000.

[14] Judith S. Dahmann, Richard M. Fujimoto, Richard M. Weatherly, "The Department of Defense Hight Level Architecture", Proceedings of the 29th Conference on Winter simulation WSC '97, December 1997

[15] Paterson D., Hougland De E., and Sanmiguel J., "A GateWay/Middeleware HLA implementation and the extra services that can be provided to the Simulation", 2000 Fall Simulation Interoperability Workshop Conference Proceedings, no. 00F-SIW-007, 2000

[16] A. D'Ambrogio and D. Gianni, "Using CORBA to Enhance HLA Interoperability in Distributed and Web-Based Simulation, " in Proceedings of the Nineteenth International Symposium on Computer and Information Sciences, Antalya, Turkey, 2004.

[17] K. Mullally, G. Hall, D. Gordon, B. Pemberton, and C. Peabody, "Open,Message-Based RTI Implementations - A better, Faster, Cheaper Alternative to Proprietary, API-Based RTIs?," in Proceedings of the Simulation Interoperability Standards Organization Spring Simulation Interoperability Workshop, Orlando,2003.

[18] K. Morse, D. Drake, and R. Brunton, "Web Enabling HLA Compliant Simulations to Support Network Centric Applications," in Proceedings of the Command and Control Research and Technology Symposium, San Diego, 2004.

[19] K. Morse, "XMSF Profile Study Group Final Report," in Proceedings of the Simulation Interoperability Standards Organization Fall Simulation Interoperability Workshop, Orlando, 2005.

[20] M. Rose, "The Blocks Extensible Exchange Protocol Core," Network Working Group, Request for Comments: 3080, The Internet Society, [Online document] Mar. 2001, [2006 July 5], Available at FTP: ftp://ftp.rfceditor.org/innotes/rfc3080.txt

[21] A. Elkins, J. Wilson, D. Gracanin, "Security Issues in High Level Architecture Based Distributed Simulation, Proceeding from the Winter Simulation Conference, 2001

[22] Jeffrey Wolff, " A secure Infrastructure for DoD High Level Architecture Distributed Simulations", University of Minnesota, 2004

[23] Filsinger, J., and H. O. Lubbes. 1996. System Security Approach for the High Level Architecture (HLA). In Proceedings of the 14th Workshop on Standards for Interoperability of Distributed Simulation (Winter).

[24] Bieber, P., J. Cazin, P. Siron, and G. Zanon. 1998. Security Extensions to ONERA HLA RTI Prototype. In Proceedings of the 1998 Fall Simulation Interoperability Workshop. Paper number 98F-SIW-086.

[25] P. Siron, "Design and Implementation of a HLA RTI Prototype at ONERA", submitted to the Fall Simulation Interoperability Workshop, 1998

[26] B. d'Ausbourg, P. Siron, " A Secure Medium Access Control Protocol: Security Versus Performances", European Symposium On Research In Computer Security(ESORICS94), 1994

[27] P. Koskelainen, H. Schulzrinne, and X. Wu, "A SIP-based Conference Control Framework," in Proceedings of the 12t h International Workshop on Network and Operating Systems Support for Digital Audio and Video, Miami Beach, 2002.

[28] J. Rosenberg and H. Schulzrinne "Models for Multi Party Conferencing in SIP," Sipping Working Group, Internet-Draft, The Internet Society, [Online document] July 2002, [2006 July 5], Available at HTTP: http://www3.ietf.org/proceedings/02jul/I-D/draft-ietf-sipping-conferencingmodels-01.txt

[29] The Internet Engineering Task Force, [Online homepage] n.d., [2006 July 5], Available at HTTP: http://www.ietf.org/

[30] J. Rosenberg and H. Schulzrinne "Models for Multi Party Conferencing in SIP," Sipping Working Group, Internet-Draft, The Internet Society, [Online document] July 2002, [2006 July 5], Available at HTTP: http://www3.ietf.org/proceedings/02jul/I-D/draft-ietf-sipping-conferencingmodels-01.txt

[31] J. Lennox and H. Schulzrinne, " A Protocol for Reliable Decentralized Conferencing," in P roceedings of the 13 th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Monterey, 2003.

[32] J. Rosenberg, "A Framework for Conferencing with the Session Initiation Protocol (SIP)," Network Working Group, Request for Comments: 4353, The Internet Society, [Online document] Feb. 2006, [2006 July 5], Available at FTP:ftp://ftp.rfc-editor.org/in-notes/rfc4353.txt

[33] Malpani, R., and Rowe, L. A. Floor control for large-scale Mbone seminars. In Proc. of ACM Multimedia (Seattle, Washington, Nov. 1997).

[34] Sisalem, D., and Schulzrinne, H. The multimedia internet terminal (MInT). Telecommunications Systems 9, 3-4 (Sept. 1998), 423-444.

[35] Schubert, I., Sisalem, D., and Schulzrinne, H. A session floor control scheme. In Proc. of International Conference on Telecommunications (Chalkidiki, Greece, June 1998).
[36] D. Andrews, J. Wharington, D. Stratton, "SexProxy – A Proposed Security Architecture for the HLA", University of Ballarat, 2001

[37] OASIS, "WS-ResourceProperties specification". 1.2 Working Draft 01. Web Services Resource Framework(WSRF) TC. OASIS. June 2004.

[38] G. Stix, The triumph of the light. 284:68.73, January 2001.

[39] Eclipse, [Online homepage] 2006, [2006 July 5], Available at HTTP: http://www.eclipse.org/platform/

[40] National Institute of Standards and Technology NIST SIP, [Online homepage] Feb 2001, [2006 July 5], Available at HTTP: http://snad.ncsl.nist.gov/proj/iptel/

[41] W. Zong, Y. Wang, W. Cai, and S.J. Turner. "Grid services and service discovery for HLA-based distributed simulation." In Proceedings of the IEEE/ACM Distributed Simulation Real Time Application Symposium, Pages 116-124, 2004

[42] Yong Xie, Yong Meng Teo, Wentong Cai, and Stephen J Turner, "Service Provisioning for HLA-based Distributed Simulation on the Grid", In Procs. of the 19th IEEE/ACM/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2005), pp.282-291, Monterey, California, USA, June 2005.

[43] XMSF Workshop position paper. http://www.movesinstitute.org/xmsf/xmsf.html.

[44] Department of Defence, Defense Modeling and Simulation Office, "RTI 1.3-Next Generation Programmer's Guide, Version 5", 2002

Appendix A - The sLCC Message DTD

This appendix shows the XML Document Data Type used for all inter-sLCC messages. This is defined in the implementation as file "LCCMsg.dtd" which must be installed on every node supporting an sLCC.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- LCCMsg.dtd defines possible LCC control messages and user announcements in
XML terms -->
 <!ELEMENT message (
                              conference,
                              reply,
                              commands?,
                              announcements?)>
        <!ELEMENT fedName EMPTY>
        <!ATTLIST fedName   federationName NMTOKEN #REQUIRED>


        <!ELEMENT dnName EMPTY>
        <!ATTLIST dnName  remoteDnName NMTOKEN #REQUIRED>

        <!ELEMENT lccTo EMPTY>
        <!ATTLIST lccTo    lccToName NMTOKEN #REQUIRED>

        <!ELEMENT lccFrom EMPTY>
        <!ATTLIST lccFrom   lccFromName NMTOKEN #REQUIRED>


        <!ELEMENT conference EMPTY>
        <!ATTLIST conference   confID NMTOKEN #REQUIRED>

        <!ELEMENT reply EMPTY> <!-- is this a reply, if yes attribute = "t", else its an
                                    initial command -->
        <!ATTLIST reply   TF NMTOKEN #REQUIRED>

        <!ELEMENT commands (command?)>
              <!ELEMENT command (fids*, sif-list?)>
              <!ATTLIST command  name NMTOKEN #REQUIRED>  <!--PI, SI etc -->

                        <!ELEMENT fids (fid+)>
                        <!ATTLIST fids id_type_list NMTOKEN #REQUIRED> <!--FID
                                + uA, oA, O or Os -->
```

```
<!ELEMENT announcements (announcement?)>
        <!ELEMENT announcement EMPTY>
        <!ATTLIST announcement   the_announcement CDATA #REQUIRED>
```

Appendix B - Simple Chat Demonstration Federation Object Model (FOM) File

The FOM, in addition to defining other information needed by an RTI, describes the data that can be exchanged between federates using two data representations: objects and interactions. They are defined in the FOM using an inheritance scheme. In this scheme, all object classes are sub-classes of HLAObjectRoot. Each sub-class object inherits the attributes of its parent. The HLAObjectRoot contains one attribute: the HLAprivilegeToDeleteObject attribute (the owner of which may delete a given object class instance).

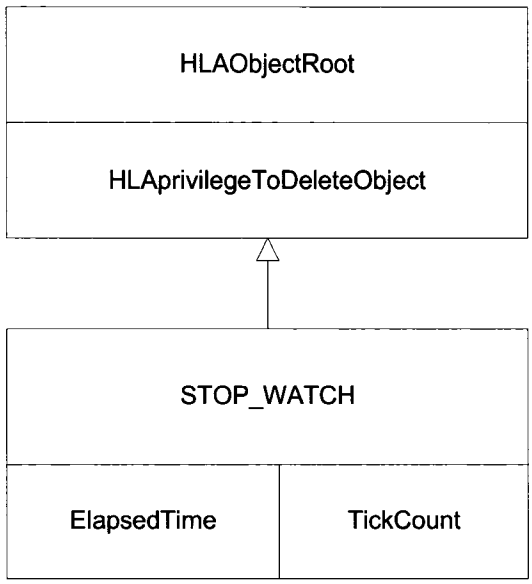

Figure B-1 Simple Chat Federation Object Class Hierarchy Diagram

The object class hierarchy may be represented in an object class hierarchy diagram; the object class hierarchy diagram for the simple chat demonstration federation is shown in figure B-1. It is derived from the FOM listed below. A FOM is an XML document and must be based on the HLA document type definition found in [11]. The demonstration federation FOM defines one object class, "STOP_WATCH", which has two attributes: "ElapsedTime" and "TickCount".

```
<?xml version="1.0"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel
        DTDversion="1516.2"
        name="Example"
        type="FOM"
        version="1.0"
        date="2006-08-16"
        purpose="FedGrid Demo FOM - I-Hsien Yeh"
        sponsor="Claude Van Ham">
        <objects>
                <objectClass name="HLAObjectRoot">
                        <attribute name="HLAprivilegeToDeleteObject"/>
                        <objectClass name="STOP_WATCH">
                                <attribute name="ElapsedTime"/>
                                <attribute name="TickCount"/>
                        </objectClass>
                </objectClass>
        </objects>
</objectModel>
```

Appendix C – Simple Chat Demonstration Federation Conference Map

The conference map shown in figure C-1 results when the FOM in Appendix B is parsed by an LRC, the required conferences created in the CI and the attendees representative of the simple chat demonstration federation are placed by calls to the SIP-RTI' s publish and subscribe services.

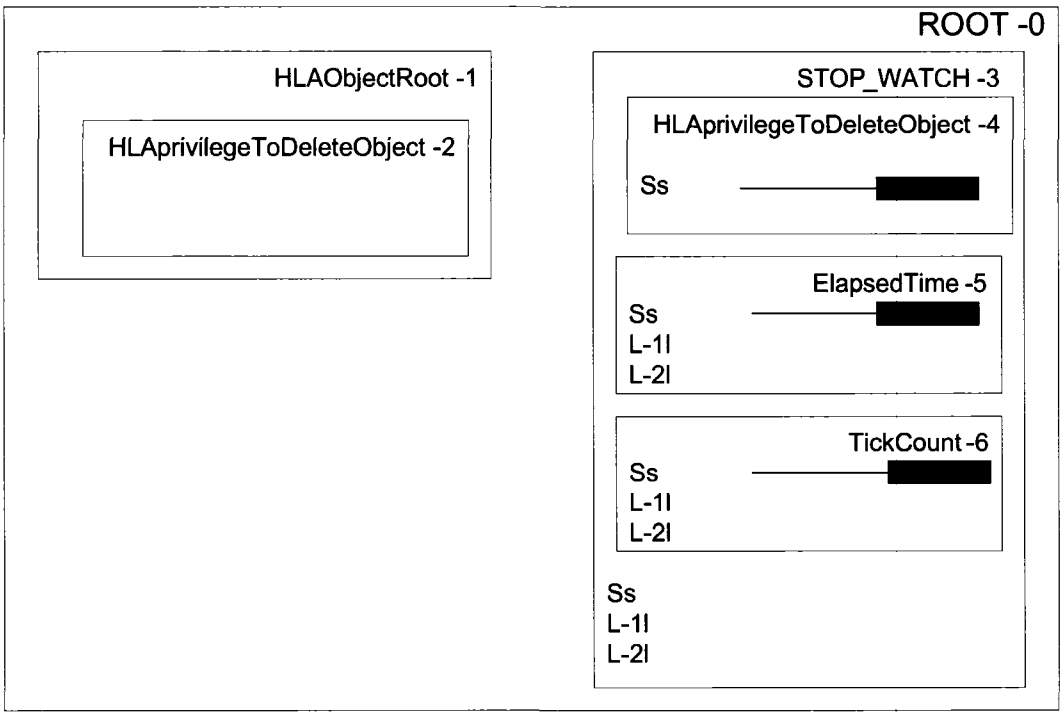

Figure C-1 Simple Chat Federation Conference Map

The SIP-RTI creates all object class conferences as "first level" user conferences, that is they are the first level accessible to users, the ROOT being accessible only to the CI. Each object class conference is parent to its attribute conferences. In this diagram 'S ,'L-1' and 'L-2' represent the speaker, listener-1 and listener-2 federates' attendees respectively. The numbers in the upper right corner of each conference box indicate its

conference ID or "confID" which is used by the CI to identify each. They are preceded by the conference names which are taken from the FOM and used by the federates in identifying the related objects or attributes when requesting an RTI handle from the SIP-RTI.