Reducing Time Cost of Distributed Run-Time Infrastructure[*]

Zhong Zhou and Qinping Zhao

School of Computer Science and Engineering, BeiHang University,
Beijing 100083, P.R. China
zz@vrlab.buaa.edu.cn

Abstract. RTI(Run-Time Infrastructure) provides services for HLA-based distributed simulation, and decides to a great extent the simulation scalability and efficiency. Distributed RTI has good scalability, but its time cost is always higher because of the architecture complexity. BH RTI is based on a distributed RTI architecture. Some techniques are used to overcome the problem of time cost. LoI-based data delivery algorithms are presented to speed up data delivery. PDU(protocol data unit)-function registry and RTI process model are designed to promote the efficiency of data packet processing. Experiment results illustrated that distributed BH RTI depressed the negative influence of architecture complexity and achieved a relatively lower time cost.

1 Introduction

High Level Architecture (HLA) is the prevailing standard of distributed simulation. It consists of three parts, framework and rules, object model template (OMT) and interface specification[1]. Run-Time Infrastructure (RTI) is prescribed to implement the interface specification. It provides services for HLA-based simulation, and decides to a large extent the simulation scalability and efficiency. The standard of HLA 1.3 has been widely used in military fields[1]. However, when HLA was approved as IEEE 1516 standard in September 2001, a lot of changes were made[2,3,4].

Some RTIs have been developed such as DMSO RTI NG[5,6], pRTI[7], MÄK RTI[8], BH RTI[9], KD-RTI[10] and starlink[11] etc.. Because HLA doesn't specify RTI implementation and interoperability, RTI design differs much. Different RTI cannot interoperate without the participation of each producers[12]. RTI architecture and performance are one of the main concerns in HLA/RTI applications.

This paper presents a distributed RTI architecture implemented in BH RTI. It supports multiple HLA standards by shielding interface differences in Local RTI Component(LRC). Then some techniques to overcome the problem of time cost in distributed RTI are presented. In the end experiment and result analysis are given.

[*] This paper is supported by the National Grand Fundamental Research 973 Program of China under Grant No. 2002CB312105.

2 Problem Initiation

There are many changes in HLA interface specification transition. Messages from SISO CFI workgroup said that next evolution of HLA standard would possibly be made in 2006 or 2007. Some means have been put forward to interoperate HLA 1.3 and IEEE 1516 federates. The Pitch AB inc. developed 1516 adapter [13]. It provides HLA 1.3 APIs based on 1516 RTI implementation. Then HLA 1.3 federates can run on pRTI 1516. The MÄK Technologies Inc. said that MÄK RTI 1.3 and MÄK RTI 1516 were built from the same code base. The two can retain compatibility between 1.3 and 1516 federates[14]. We developed RTIBridge [16] based on bridge federate[15] to link 1.3 and 1516 federates. RTIBridge can link federates between HLA 1.3-based BH RTI, DMSO RTI NG and IEEE 1516-based starlink. RTIBridge covers only part interfaces, and is only suitable for small scale simulations because of the bottleneck of bridge federate.

Unpredicted revision may be made to HLA standard in future. There exist several versions of HLA standard. These problems bring much work for interoperation, other than transplant or modification. The trouble also exists in code preservation and tedious test procedure. In this background, the interoperation problem should be taken into consideration in further RTI development.

3 Distributed RTI Architecture

RTI implementation has no fixed architecture. Current RTI architectures mainly have three types: central, distributed and hierarchical, according to the relationship of CRC (Central RTI Component) and LRC. DMSO RTI is the first RTI, and had been published for several years. It has a great influence on the field. A large portion of current RTIs are based on a traditional DMSO RTI like architecture as Figure 1. Considering the interoperation of heterogeneous systems, we made an effort in distributed RTI architecture and developed BH RTI.

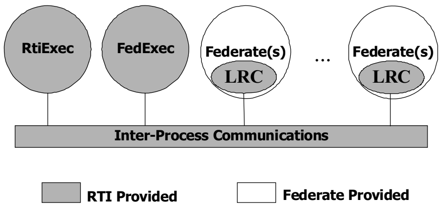


Fig. 1. Traditional DMSO RTI like architecture

The distributed RTI architecture owns several RTI nodes in peer to peer. Each RTI maintains only the required data to serve connected federates. BH RTI is divided into two processes, LRC and rtiexec (Figure 2). LRC provides a programming library following the standard. It is responsible for interface implementation based on the rtiexec. Process rtiexec is designed to provide common services of distributed simulation for LRC. The standard differences are shielded in RTI-LRC services. Then minor modification is required when the standard changes. New LRC can be developed based on RTI-LRC service for a revised HLA standard or even another standard. With this distributed RTI architecture, much work on the standard transition can be cut off, and federates of different standard are easier to interoperate.

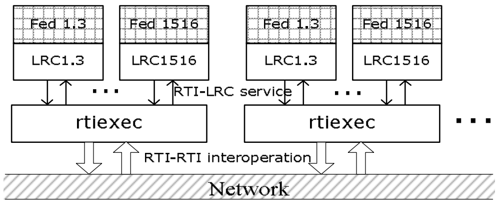


Fig. 2. Software Architecture of Distributed RTI

Although distributed architecture is good at scalability, time cost is a main negative problem for more layers and architecture complexity. Liu compared time cost of three RTI architectures[11] in which the time cost of distributed RTI is narrated as

T(distributed)=T(sender_localLRC)+T(globalComputing)+T(localLRC_remoteLRC) + T(remoteLRC_receiver)

The time cost of distributed RTI is higher than the other two with the comparisons. To be more precise, the time cost of this distributed RTI architecture should be

T(distributed)=T(sender_localLRC)+T(globalComputing)+T(localLRC_RTI)+T(sender_rti)+T(RTI_RTI)+T(remote_RTI)+T(remoteRTI_LRC)+T(remoteLRC_receiver)

The constitution of total time cost is rather complicated, including three network cost factors, T(localLRC_RTI), T(RTI_RTI), T(remoteRTI_LRC), together with four process cost factors. However, this is just analysis in theory. Each part of the cost may vary much in different implementations. The following parts present techniques applied in BH RTI to overcome the problem of time cost.

4 LoI-Based Data Delivery Algorithms

Data delivery in HLA requires attribute set matching for publish/subscribe. Sometimes region overlap is also required when DDM is used. Existing publish-subscribe mechanisms can only judge whether a message is relevant to a subscriber or not. Aiming to solve the relevance evaluation problem, a new relevance evaluation mechanism Layer of Interest (LoI) was proposed in our previous work [17][18]. LoI defines a relevance classifier based on the influence of spatial distance on receiving attributes and attribute values. Some related LoI variables are listed in Table 1. Based on LoI, new data delivery algorithms are presented in this section.

In our recent research, we drew two important deductions about the LoI relationship among the three parts, LoIs of the publisher, the subscriber and messages.

Table 1. Symbols in the LoI Mechanism

Symbol	Definition
$P_m^{(i)}$	LoI of publisher over object class ***i*** with ***m***-size attribute set
$p_m^{(i,o)}$	LoI of local object instance ***o*** of object class ***i*** with ***m***-size attribute set
$\eta_j^{(i,o)}$	LoI of attribute update/reflect with ***j***-size attribute set of object instance ***o*** of object class ***i***
$S_k^{(i)}$	LoI of subscriber over object class ***i*** with ***k***-size attribute set
$s_l^{(i,o)}$	LoI of remote object instance ***o*** of object class ***i*** with ***l***-size attribute set

Deduction 1. A publisher can only send attribute updates of LoI $\eta_j^{(i,o)} \le p_m^{(i,o)}$.

Deduction 2. A subscriber can only receive attribute reflects of LoI $\eta_j^{(i,o)} \le s_l^{(i,o)}$.

In the new publish-subscribe environment, the publisher works with $P_m^{(i)}$ of object class ***i*** and $p_m^{(i,o)}$ of local object instance ***o***. And a subscriber works with $S_k^{(i)}$ of object class ***i*** and $s_l^{(i,o)}$ of remote object instance ***o***. The four LoIs denote the dynamic detail relevance in publish-subscribe sides. Messages will be tagged with LoI $\eta_j^{(i,o)}$. $\eta_j^{(i,o)}$ represents the fundamental relevance of messages. Then new data delivery algorithms below can be obtained, according to Deduction 1 and 2.

Algorithm 1. (Algorithm for sending data) LoI_UAV

```
FOR each attribute update of local object instance o
  int l = $\eta_j^{(i,o)}$, compute according to the Definition.
  IF (l ≤ $p_m^{(i,o)}$)
    //attach l to the update packet;
    update.loi = l;
  multicast the update packet to the subscriber group;
END FOR
```

Algorithm 2. (Algorithm for receiving data) LoI_RAV

```
FOR each attribute reflect of remote object instance o
  int l = reflect.loi; //get $\eta_j^{(i,o)}$
  IF (l ≤ $s_l^{(i,o)}$)
    //the reflect packet is wanted by the subscriber
    accept the reflect packet;
  callback the corresponding user function;
END FOR
```

Complex and costly attribute set matching is simplified, and the data sending and receiving become more efficient. Receivers can also perform precise attribute set matching as Algorithm 3.

Algorithm 3. (enhanced receiving data) PreciseLoI_RAV

```
FOR each attribute reflect {HVP_j{<attr_i, value_i>}, loi} of remote object instance o
  int l = reflect.loi;
  IF (l ≤ $s_l^{(i,o)}$)
  WHILE(int h = 1; h <= j; h++){
    IF attr_h ∉ subscription attribute set {c_i, <attr_i>}
      remove <attr_h, value_h> from HVP_j{<attr_i, value_i>}; }
  IF (sizeof HVP_j{<attr_i, value_i>} = = 0)  break;
  accept the reflect's remaining data HVP_j{<attr_i, value_i>};
  callback the corresponding user function;
END FOR
```

LoI-based data delivery algorithms were applied into BH RTI. Figure 3(a) is the sending process. Figure 3(b) is the receiving process. Four processes are included, HLA application (federate), LRC, RTI and other RTIs. Here RTI means rtiexec of BH RTI. BH RTI provides services for federates by LRCs, each LRC for one federate.

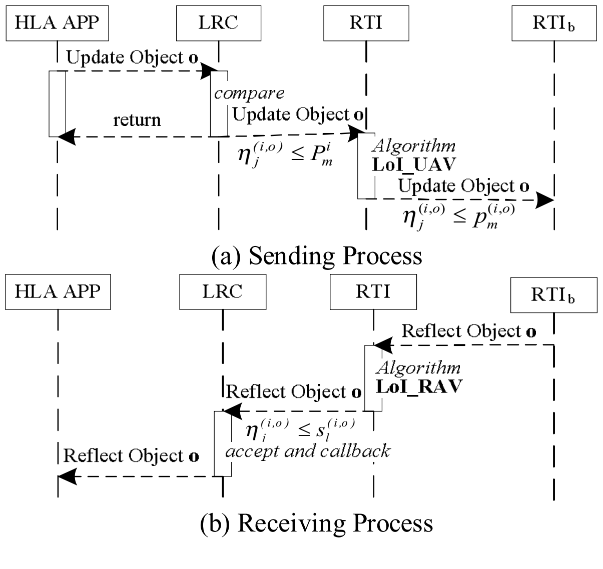


Fig. 3. LoI-based Data Delivery Process in BH RTI

5 PDU-Function Registry

BH RTI involves two types of inter-process communications (IPC), RTI-LRC service and RTI-RTI interoperation. RTI-LRC service communication is using TCP. RTI-RTI interoperation is based on multicast UDP, using multicast to filter irrelative messages. We define a set of protocol data units (PDUs) for each communication, inter-PDU for RTI-LRC service and outer-PDU for RTI-RTI interoperation.

PDU is made up of PDUheader and PDUdata, illustrated in Figure 4. The PDU-header comprises PDU version, type, length, time stamp etc.. The PDUdata uses union data structure, which stores data of each PDU. The length in PDUheader is calculated by adding up the actually used size of PDUdata to the size of PDUheader. That is, it indicates the actual space requirement of a PDU (Figure 4). In this way only the useful data of PDU is communicated to save the bandwidth and transportation time. When a PDU is arrived, RTI will extract the PDU type from the header and pick up data from PDUdata according to the PDU type.

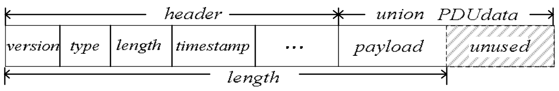


Fig. 4. PDU Structure

Distributed architecture has more layers and complicated structure. The PDU design can only simplify the coupling of IPC. Some methods should be taken to simplify the modules' coupling and invoking. BH RTI uses multiple threading to enhance the efficiency of services, but too many critical sections may induce efficiency decrease or dead lock. Aiming to speed up data packet processing, a mechanism of PDU-Function registry for data process is designed on the thought of callback.

PDU has fixed format to parse, so the declaration of process functions can be fixed. First, we bind a PDU type to the memory address of corresponding functions. Second, when the process thread picks up a PDU of this type, it looks up in the registry for function addresses. Then the process functions can be invoked directly(Figure 6). There may be access to other services or data in the body of function. So a pointer to the services or data is attached to the registry while binding. Then addresses requiring critical sections in the functions can be easier to check.

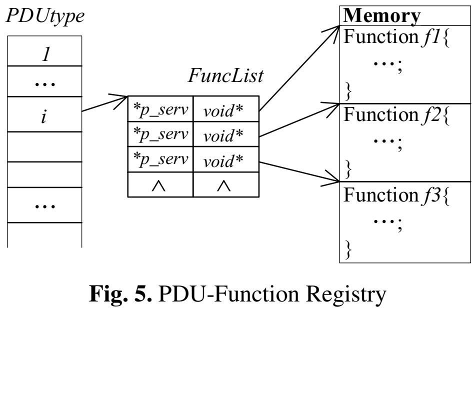


Fig. 5. PDU-Function Registry

The pseudo code of processing an interPDU is as the following. The relationship between adjacent layers of distributed architecture is simplified with this method.

```
interPDU pdu = receiveNextPDU();
int type = pdu.header.type;
KernelService* p_service = getServicePTR();
IF( funcList(type)==NULL || funcList(type).totalcount==0)
  RETURN;
func* p_func = funcList(type).first();
LOOP{
(*p_func)(&pdu, p_service);
p_func = funcList(type).next(p_func);
} UNTIL(p_func == NULL);
```

7 RTI Process Model

The time to callback user code is an important issue. HLA standard doesn't specify exactly how RTI process model should work. Three types of RTI process model are used in current RTIs: single-threaded, asynchronous and multi-threaded[19].

BH RTI implements single-threaded and multi-threaded process model. User can select either by configuration. BH RTI processes PDU queue of LRC according to the configurations in Figure 6. (a) When the single-threaded model is used, LRC puts received PDUs into queue and the federate executes the callbacks to the queue in tick(). (b) In the multi-threaded model, tick() is unnecessary and the process PDU thread will execute the callbacks after an arrived PDU is valid to be processed.

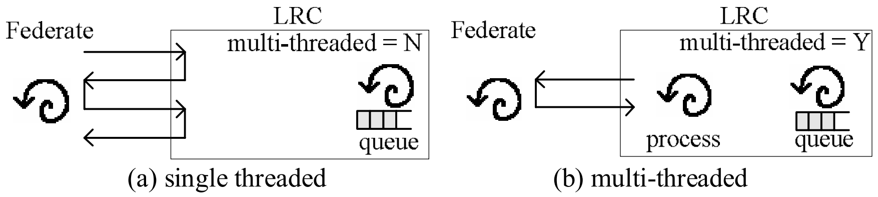


Fig. 6. Implementation of BH RTI Process Model

There are advantages and disadvantages for each process model. Single-threaded model has no thread switching, but the frequency and occasion of invoking tick() are often puzzles. It's easy to use an asynchronous model because developers needn't think about occasion for tick(). But the specific time is required to be waited no matter whether there is a callback. It's efficient to use multi-threaded model, but developers are required to treat with the threading safety.

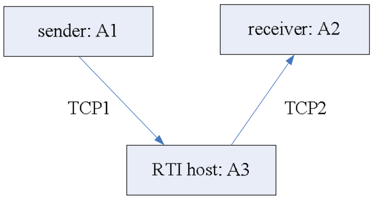
8 Experiment Evaluation

Two experiment results are presented in this section, after introduction to the experiment of pure TCP/UDP time cost. First, we investigate the time cost constitution of BH RTI to analysis the factors. Second, the delay comparison of some commonly used RTIs is performed. The experiments are conducted in network using Huawei Switch Quidway S3050. The detailed setup is shown in Table 2.

Table 2. Host Setup for Experiment

Host Id	CPU	RAM	OS	NetworkCard
A1	P4 2.8G	512M	winXP	10/100M
A2	P4 3.0G	512M	winXP	10/100M
A3	P4 2.4G	768M	winXP	10/100M

The setup for experiments is shown in Figure 7 for traditional DMSO RTI like RTI experiments and Figure 8 for distributed RTI experiments.



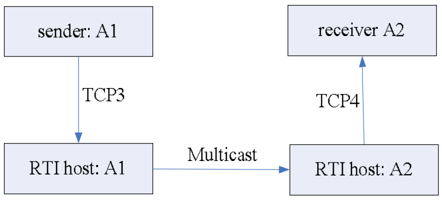


Fig. 7. Setup for Central RTI Experiment **Fig. 8.** Setup for Distributed RTI Experiment

Before the two experiments, the variation of data delivery in pure TCP/UDP links is measured for reference. We have measured the time cost of all the links including the TCP of A1 to A3, TCP of A3 to A2, A1's multicast, localhost TCP of A1, localhost TCP of A2 used in Figure 12 and 13. Figure 9 is the results of time cost in pure TCP/UDP links with different payload.

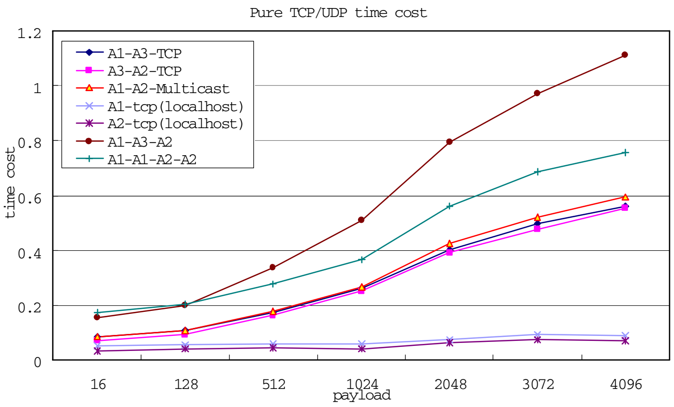
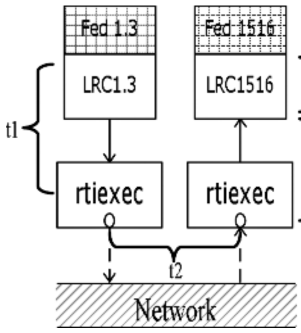


Fig. 9. Pure TCP/UDP time cost

To better study the time cost constitution, we divide the time cost of BH RTI into 4 parts as Figure 10. T(BH RTI) = t1 + t2 + t3 + t4. We test the 4 factors carefully by setting check points in code. Figure 11 presents the time cost constitution of BH RTI 2.2 using the setup in Figure10. We can see that time cost of A1's localhost TCP subtracted from t1 is below 0.05ms. And t2 or t3's subtracting corresponding delivery time cost is below 0.1ms. Factor t4 is nearly 0.03ms. This comparison denotes that each factor in BH RTI's time cost is rather small.



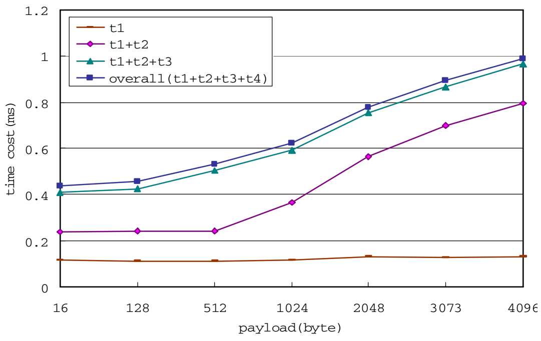


Fig. 10. Time cost experiment of BH RTI **Fig. 11.** Time Cost Constitution of BH RTI 2.2

The RTI delay experiment results is as Figure 12, in which BH RTI 2.2(Central mode), DMSO RTI 1.3NGv6, pRTI 1516v2.3 is using the setup of Figure 7 and BH RTI 2.2 is using the setup of Figure 8. The central mode of BH RTI refers to that several federates connect to a unique BH RTI for RTI services. From the experiments results, we can see that BH RTI 2.2 has comparatively smaller time cost. The distributed mode of BH RTI 2.2 is premier in the time cost in the experiments. The central model of BH RTI 2.2 is relatively smaller in small payload, and there's an increase when the payload adds because of the increase of time cost in heavy payload.

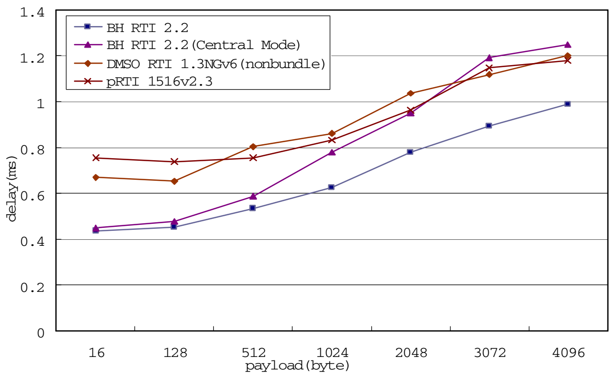


Fig. 12. Delay Comparison of RTIs

The results of both experiments give us an idea that different factor of time cost may have different weight. The time cost of different architecture cannot be compared by simple adding up. The weight of factors should be taken into consideration. And the architecture is not the only dominating factor in time cost. The concrete design and techniques also contribute much to the time cost.

9 Conclusion

Distributed RTI architecture has good scalability and comparatively higher time cost in theory. A distributed RTI architecture used in BH RTI is presented in this paper. Some key techniques to overcome the problem of time cost are introduced, including data delivery, data packet process and RTI process model. Experiment results illustrated that the distributed BH RTI overcame the negative influence of architecture complexity and had a relatively lower time cost. Techniques narrated in this paper have been evaluated in practical applications, some of which may be useful to the development of other distributed systems.

References

1. Defense Modeling and Simulation Office, High Level Architecture interface specification version 1.3, April 1998.
2. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) – Framework and rules. (IEEE Std 1516-2000). New York:The Institute of Electrical and Electronics Engineers Inc., 2000.
3. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) – Federate interface specification (IEEE Std 1516.1-2000). New York:The Institute of Electrical and Electronics Engineers Inc., 2000.
4. IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification (IEEE Std 1516.2-2000). New York:The Institute of Electrical and Electronics Engineers Inc., 2000.
5. S.Bachinsky, J.Noseworthy, F.Hodum, Implementation of the Next Generation RTI, Spring Simulation Interoperability Workshop, Orlando, FL., USA, 1999.
6. Defense Modeling and Simulation Office, Department of Defense. 2002. RTI 1.3 – Next generation programmer's guide version 6. http://www.dmso.mil/.
7. Mikael Karlsson, Lennart Olsson, pRTI 1516- Rationale and Design. Fall Simulation Interoperability Workshop, Orlando, FL., USA, 2001.
8. Douglas D Wood, Len Granowetter. Rationale and Design of the Mak Real-Time RTI. Spring Simulation Interoperability Workshop, Orlando, FL., USA, 2001.
9. BH RTI 2.3 User Guide. http://www.hlarti.com/
10. Hao JG, Huang H. Implementation architecture of KD-RTI. System Modeling & Simulation, 2002,1(1):48-52.
11. Liu Buquan, Wang Huaimin, Yao Yiping. Key techniques of a hierarchical simulation runtime infrastructure—StarLink. Journal of Software(in Chinese), 2004, 15(01): 9-16
12. Michael D Myjak, Duncan Clark, Tom Lake, RTI interoperability study group final report. Fall Simulation Interoperability Workshop, Orlando, FL., USA, 1999.
13. 1516 adapter for HLA 1.3 federates, http://www.pitch.se/1516adapter/default.asp
14. Mak high performance RTI, http://www.mak.com/products/rti.php

15. Juergen Dingel, David Garlan, Craig Damon. Bridging the HLA: problems and solutions. Sixth IEEE International Workshop on Distributed Simulation and Real Time Applications Fort Worth, Texas, USA, 2002
16. Cai Nan, Zhou Zhong, Wu Wei. Research on the interconnection of heterogeneous RTIs and multi-federations based on Bridge Federate. Journal of Computer Research and Development. to be appeared in vol.43, 2006
17. Zhou Z, Zhao QP. Extend HLA with layered priority. In: Proceedings of the Spring Simulation Interoperability Workshop. Orlando FL, 2003. Paper 03S-SIW-012.
18. Zhou Z, Zhao QP. Research on RTI congestion control based on the layer of interest. Journal of Software, 2004,15(1):120~130.
19. Mikael Karlsson, Peter Karlsson, An in-depth look at RTI process model, Spring Simulation Interoperability Workshop, Orlando, FL., USA, 2003.