网格环境下基于 HLA 的分布交互仿真动态交互研究*)

窦志武　邓贵仕

（大连理工大学管理学院系统所　大连 116023）（沈阳炮兵学院　沈阳 110162）

摘　要　基于 HLA 开发的分布交互仿真缺乏灵活性，不支持动态资源分配。网格服务是开发基于网络的分布交互仿真的另一途径，但只应用网格服务开发分布交互仿真，开发者必须自己创建通信机制来协调仿真执行。本文提出了一个动态交互框架，将基于 HLA 的分布交互仿真建立在网格服务的基础之上，来克服两者所存在的问题并发挥各自的优点，实现仿真资源的动态分配、动态状态监控及任务动态迁移。文中详细地阐述了框架的构成、框架关键服务构建的设计和实现、各种网格服务的机制及服务接口，并给出实例进行验证。

关键词　网格，网格服务，分布交互仿真，动态交互

HLA－based Dynamic Interactive of Distribute Interactive Simulation under Grid Environment

DOU Zhi-Wu　DENG Gui-Shi

(School of management, Dalian University of Technology, Dalian 116023)

(Shenyang Artillery Institute, Shenyang 110162)

Abstract　The HLA-based simulation is short of flexibility and does not support dynamic resource assign ation. Grid service is another method to devedop distribute interactive simulation based on network, only using grid service to develop distributed simulation, and developer had to establish communication mechanism to make simulation smooth by self. In this paper, a dynamic structure framework is proposed. The framework develops distribute interactive simulation based on HLA under grid environment to overcome the defects of both technologies and to resolve the problems of dynamic distribution of resource , dynamic monitoring of situation and dynamic migration of tasks. The components of the framework, the mechanism of various services and the implement of interface are discussed in detail, and an experiment was provided in the end of the paper.

Keywords　Grid, Grid service, Distributed interactive simulation, Dynamic interactive

1　引言

高层体系结构(High Level Architecture, HLA)目前是开发基于网络的分布交互仿真的最流行的 IEEE 建模与仿真标准[1~3]。但在 DMSO RTI 中[4]，发现控制过程 RTIexec 是通过在配置文件中明确描述端点实现的，这种静态的本质缺乏灵活性，使 HLA 不支持资源动态分配、动态状态监控及任务动态迁移。

网格服务[5](Grid Service)是构建分布交互仿真的另一种途径。网格服务是 Web 服务的扩展，Web 服务通过设计软件系统来支持网络上计算机与计算机之间的交互。它提供一组接口，这些接口被明确定义并遵守特定的惯例，解决服务发现、动态服务创建、生命周期管理、通知等。单独网格和 Web 服务来构建分布交互仿真也存在问题，开发者必须创建自己的通信机制来协调仿真执行，这是个非常繁琐而困难的工作[6]。

因此本文将建议一个网格环境下基于 HLA 的分布交互仿真动态交互框架，扬长避短来解决两者在开发分布交互仿真中所存在的问题。

2　网格服务在分布交互仿真中的应用研究

多年来，分布交互仿真的资源管理吸引了众多研究者的注意。文[7]提出了一个资源共享系统，用来解决基于 HLA 的分布交互仿真中的负载平衡问题。利用 GT3 的网格服务来构建分布交互仿真的方法在文[8]中进行了详细的阐述，该方法利用一个服务器负责仿真资源的分配，实现了仿真资源与仿真应用的分离。该模型的不足是开发者必须亲自负责通信机制的定义和开发。另一有趣的方法结合了 Web 技术和 HLA 技术，在文[9]中提出。在这一方法中，RTI 调用通过 SOAP 和 BEEP 通信层来实现，这样可以通过 RTI 实现双向的调用和回调。

目前研究的热点是如何集网格技术和 HLA 的优点于一身来开发分布交互仿真。资源管理问题在文[10]中进行了论述，为基于 HLA 的分布交互仿真建议了一个负载管理系统，该系统建立在网格环境之上。文[11]提出的系统支持在网格环境下运行基于 HLA 的分布交互仿真，通过网格管理系统实现联邦迁移。这种方法与本文提出的方法的关键不同在于，前者必须提供 HLA 仿真代码，同时用户可以利用后者发现的仿真模型来构建大规模的仿真。为基于 HLA 仿真应用构建网格服务的三层方法在文[12]中提出。文[13]提出了网格环境下基于 HLA 的分布交互仿真的网格服务发现的框架和过程。这个框架与本文提出的方法的关键不同在于，前者只是用来发现模型并构建仿真，而后者是从动态交互的角度来探讨仿真过程中仿真资源的动态分配、动态状态监控及任务动态迁移。

3　网格环境下基于 HLA 分布交互仿真的动态交互框架

在本文建议的框架中，核心内容由 3 部分组成：核心信息服务(central information service)、联邦临时注册服务(Feder-

*)国家自然科学基金项目：70272050。窦志武　讲师、博士生，研究方向：复杂系统建模与仿真；邓贵仕　教授、博士生导师，研究方向：信息系统工程。

ate Factory Services)和 RTI 临时注册服务(RTI factory service)。核心信息服务负责动态地监视和维护网格服务所需的资源信息、仿真程序运行状态评估、新节点的发现与代码迁移,采用这种方式,各种分布的、对用户隐藏的计算资源能够通过网格服务进行动态的配置和交互,同时动态发现 RTIExec 进程和仿真模型(也就是网格服务)并动态构建仿真联邦、平衡负载。联邦临时注册服务负责动态地注册仿真联邦实例;RTI 临时注册服务负责动态地注册 RTI 实例。利用该框架,终端用户可以从完全自己实现仿真代码中解放出来,通过组装不同的服务来动态构建大规模的分布交互仿真,也不必关心它们的仿真在哪里运行,并可以根据动态信息进行动态交互,控制仿真高效运行。该框架的潜在需求是将资源管理和安全机制集成在一起,形成成熟的完备的系统。TIExec 进程通过 RTI 服务来管理,仿真模型被封装在联邦服务里。联邦服务通过注册临时联邦服务来实现。

图 1 是本文所建议的在网格环境下执行基于 HLA 的分布交互仿真的动态交互框架。在该框架下,不同地域、不同类型、不同组织的联邦可以由联邦临时注册服务动态创建并动态管理。

信息服务:信息服务是动态交互的信息服务器,主要完成下列任务:动态保存联邦成员与 RTI 服务句柄之间关系的记录;监视、保存仿真资源的动态使用状态;仿真代码的迁移;支持来自联邦用户的查询;支持基于 RTI 的状态更新。因此它为客户、RTI 服务和联邦临时注册服务分别提供了一系列的操作。利用信息服务,不同的模型可以动态定位。它是一个全局服务,所以客户能够直接用它来求解问题。

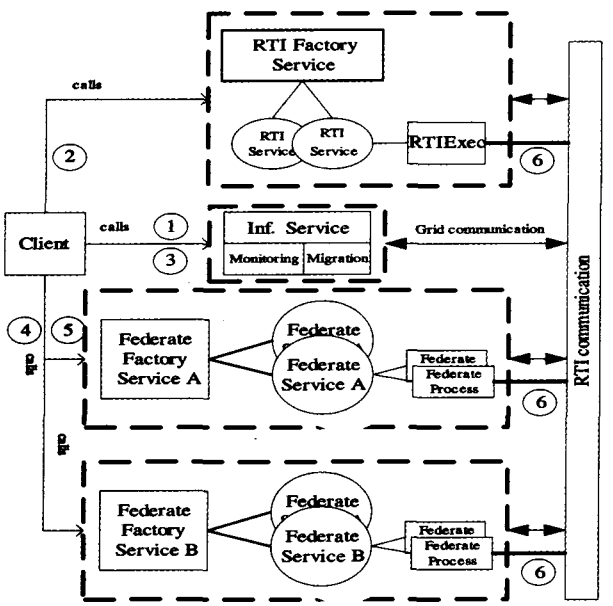

图 1 网格环境下基于 HLA 分布交互仿真动态交互框架

RTI 临时注册服务:RTI 临时注册服务动态创建 RTI 服务实例。RTI 服务实例负责向信息服务提供动态交互所需的联邦执行及 RTI 进程生命周期中的状态信息。RTI 服务实例的创建有两条途径:一是通过管理计算资源的网管注册;一是通过客户请求,当客户需要时,客户从信息服务获得 RTI 临时注册句柄后发出请求。每一个 RTI 服务监视一个 RTIExec 进程和负责仿真联邦执行管理的 daemon。RTIExec 通过请求触发,并且为了节省系统资源,当它超过一定的时间不用后,会被 RTI 服务动态撤销。当 RTI 服务实例创建后,它通过信息服务注册自己;同样,当它被撤销后,它会通过信息服务撤销自己的注册。通过监视过程 RTI 服务还负责接收 RTIExec 进程的动态信息,包括联邦成员数、成员名及相应的被信息服务注册和撤销注册的成员。作为一般规则,谁创建的 RTI 服务,谁负责撤销它。

联邦临时注册服务:联邦临时注册服务负责动态创建联邦服务实例。联邦服务实例可以接受来自客户的请求,例如创建联邦进程、返回联邦运行状态或者终止联邦进程。每一个联邦服务都与一类联邦资源紧密相连。这意味着联邦服务应该拥有足够的创建、监视和终止一定联邦执行的知识,通过联邦服务可以实现联邦资源的动态交。从技术角度来看,这一网格服务实际为用户提供了一种新的控制在远程服务器上运行的联邦执行的方法。此外,假如一个联邦临时注册服务提前在全局信息服务上注册了自己的实例,它还可以用来发现可能的联邦服务。更进一步讲,假如一个联邦服务的行为对另一网格联邦服务是一个客户,它可以创建另一服务提供的联邦执行,也就是说在没有真正的客户参与的情况下,一个联邦服务通过网格能够调用许多其它联邦服务来动态构建大规模的分布交互仿真,并根据运行情况自主进行资源动态交互及负载平衡。在后面的部分将要阐述接口方法。

这些服务的设计只是在它们的接口中提供原始的操作。通过这种方式,更高层的算法可以在这些原始的操作基础上实现,降低与一些特殊应用的服务的耦合度,提高了应用的灵活性。利用本文建议的方法,一个典型的仿真动态交互过程可以被实现(参考图 1):客户查询信息服务,为联邦成员获得它感兴趣的 RTI 服务句柄(step1),并且联系 RTI 服务去得到 RTIExec 进程所需的组播地址(step2)。客户再次联系信息服务,获得联邦临时注册服务句柄(step3),并且请求创建联邦服务实例(step4)。然后客户通知联邦服务,创建仿真联邦(step5),带有组播地址的联邦成员作为参数传递给联邦服务。创建的联邦利用组播地址来连接相应 RTIExec 进程(step6),并执行真正的仿真。依靠实际的仿真,客户也可以将为 RTI 服务查询信息服务并获得组播地址的任务留给联邦,它们有能力独立完成这些任务(图中双向箭头所示)。

4 动态交互的关键组件设计与实现

在这部分中,本文将详细研究前面所述的 3 种服务的细节,检验与其相关的设计与实现结果。

4.1 动态交互联邦成员命名方案

在开放的计算环境中,客户使用本文建议的动态交互框架同时执行不同的分布交互仿真的结果是不可预测的。这里潜在的一个问题是客户趋向使用同一个成员名执行不同的仿真。在这种情况下,仿真会出现错误。因为联邦趋向加入相同的成员,当加入同名但不同的仿真成员时,会导致仿真交互的错误,从而使仿真结束。出现这种情况的原因在于信息服务假设联邦成员名是独一无二的。作为解决这一问题的方法,本文提出了一种命名方案。在成员正常名字前加一个前缀,在该框架下所看到的前缀加正常成员名是独一无二的。前缀可以是对领域有意义的,也可以是任何可以使成员名独一无二的字符。运用该方案,用户在真正运行仿真应用前需要形成一逻辑"领域"。

4.2 动态交互信息服务

信息服务不仅是动态交互的核心信息服务器,而且担负动态交互过程中资源的动态分配、动态状态监控及任务动态迁移,这也是动态交互的 3 大主要任务。所有其它服务根据创建、更新及执行进程的需要动态向信息服务注册各种信息,信息服务根据注册的信息进行动态交互。接口如下:

```
Void registerRtiFactory(String rtiFacHandle)
Void dergisterRtiFactory()
String getRtiFacHandle()
String getRtiServiceHandleForFed(String fed)
Void registerRtiService(String rtiHandle)
Void deregisterRtiService(String rtiHandle)
Void registerFedWithRtiService(String fed,
                               String rtiHandle)
Void deregisterFedWithRtiService(String fed,
                                 String rtiHandle)
Void registerFedFactory(String facName,
                        String fedFacHandle,
                        String ver)
Void deregisterFedFactory(String facName,
                          String ver)
String getFedFacHandle(String facName,String ver)
String getFedPerformanceParameters(String Federationname,
                                   String Federatename)
String findWellPoint()
Void transportFed(String FedfacHandle,
                  String FedPerformanceParameters,
                  String WellPoint)
```

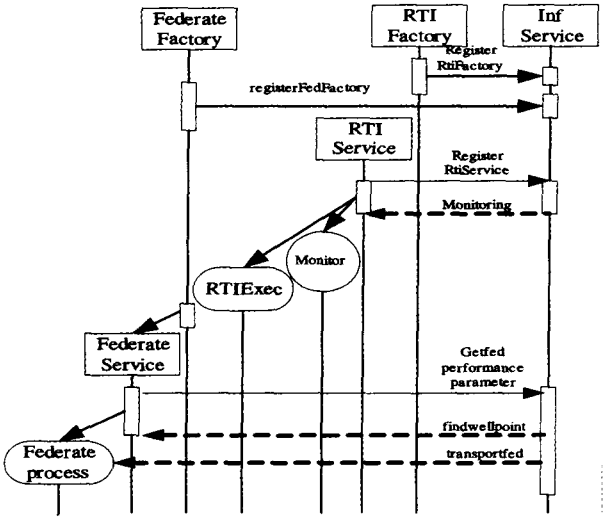

图 2　仿真中代码迁移的交互过程

图 2 描述了仿真过程中实现代码迁移交互网格服务所起的作用。系统可由用户或自动请求联邦临时注册服务和 RTI 临时注册服务，注册联邦服务及 RTI 服务，并将注册信息提交给信息服务；RTI 服务创建 RTIExec 执行进程并向信息服务注册信息，信息服务开始对进程进行监视；联邦服务创建联邦并注册信息，信息服务对仿真联邦的运行状态及仿真资源动态进行监视和评估。当发现仿真运行状态异常或节点负载异常时，启动迁移功能，首先终止该异常进程并记录当前状态参数，同时寻找负载最小的节点，然后将代码及状态参数传递到新节点，重新启动仿真。

为了避免交互中产生错误，还有一种情况必须考虑，现在以一化工生产仿真为例。假设某一未被创建的联邦成员的 RTI 服务被查询。注意，RTI 服务只有检测到联邦成员才为其注册，但这时信息服务还没有生产仿真中这一成员的任何信息。在这种情况下，一种可能的解决办法就是武断返回一 RTI 服务句柄，最适宜的就是负载最轻（当前有最少联邦成员数）的 RTI 服务的句柄。该方法简单但容易引起问题。假设这样一种情况：生产仿真中的联邦 A 和联邦 B 几乎同时发出查询这一联邦成员 RTI 服务句柄的请求。A 的请求首先被处理，返回具有最轻负载的句柄 rtiX，再处理 B 的请求，这时负载情况可能发生了变化，信息服务返回给 B 的可能是另一个句柄，如 rtiY。当这些发生后，那些趋向加入相同联邦成员的联邦将会结束与不同 RTIExec 进程的通信，导致仿真的错误改变，并且最终将使仿真结果出错。

为解决交互出现这样的问题，我们采用了一种保存机制，该机制建立一返回句柄保存数据库，并且为同一联邦成员的后续查询返回同一句柄。流程图 3 显示了这一保存机制的工作过程。

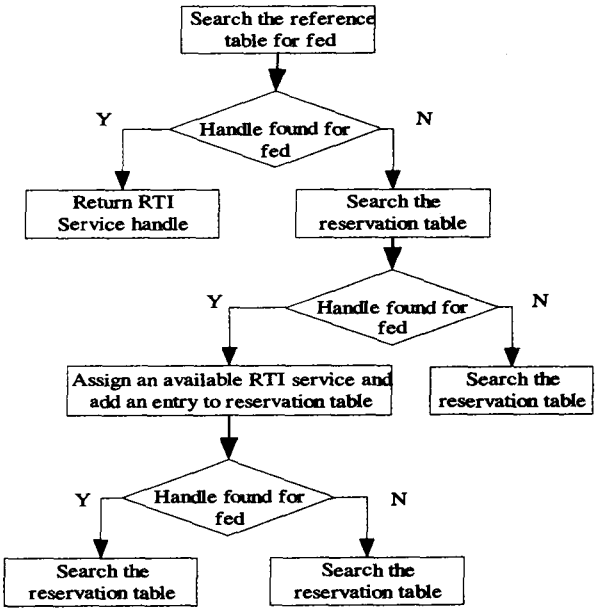

图 3　保存机制流程图

仿真过程中动态交互的信息主要靠信息服务来提供。随着信息服务器中信息的动态变化，信息服务器会动态地向 RTI 发出指令，引发不同的交互，实现进程的动态迁移，平衡负载。其作用的结果类似于移动 Agent，但实现比移动 Agent 简单易行，同时能对资源进行动态管理。

4.3　动态交互 RTI 临时注册服务

动态交互 RTI 临时注册服务的唯一目的就是创建 RTI 服务，RTI 服务完成真正的工作。仿真交互中，RTI 服务的功能是触发 RTIExec 进程，并在该进程的整个生命周期中监视其执行。RTI 服务要么没有运行的 RTIExec 进程，要么仅仅只有一个运行的 RTIExec 进程。不同的 RTI 服务负责分离的、带有不同地址的 RTIExec 进程运行，它们的 IP 地址和端口数都是随机产生的。地址的随机产生防止产生地址冲突 RTI 服务。由一个分离的线程来监视 RTIExec 进程的执行：当它发现有新的联邦成员运行在这个进程时，它会向信息服务注册这一成员；相似地，当它发现成员被撤销时，它会向信息服务撤销这一成员的注册。为了优化仿真资源利用，这一线程也监视 RTIExec 的空闲时间，并在一个预先定义的空闲时间之后终止它已释放占用的仿真资源。RTI 服务支持对 RTIExec 进程终点的查询和创建该进程的请求。接口如下：

```
Boolean isRtiRuning()
String getMmultcastEndpoint()
String creatRti()
Boolean createRtiWithEP(String endpoint)
```

信息服务器通过这些 RTI 服务注册的信息对仿真的运行状态作出评估，然后由迁移服务实现仿真代码的动态迁移、仿真资源的动态分配。

注意，CreatRti 和 CreateWithEP 实际上是同一操作的两个版本。赋予它们不同的名字是为了它们的不同用法。前者不带参数，是简单请求 RTI 服务创建一个 RTIExec 进程，可以用任何需要的地址，并且用户可以调用 getMulticastEndpoint 找回它。相反，后者用指定的地址要求 RTI 服务创建 RTIExec 进程并返回一个标志，来提示这一地址的进程是否

成功创建。这种不同允许在不同构想下灵活运用。

4.4 动态交互联邦临时注册服务

联邦临时注册服务创建联邦服务实例。通过联邦服务，客户可以初始化一个联邦进程，并由联邦服务管理。实例创建后，客户可以与联邦服务通信来远程控制联邦的运行，与信息服务及 RTI 服务结合，实现仿真过程中的动态交互。

联邦服务包含 4 种方法可以为用户调用，分别是 createFederate，createFederateWithRti，getFederateStatus 和 terminateFederate。这 4 种接口的方法如下：

```
Boolean creataFederate(String federationName,
String federateName,
String[] federateParameters)
Boolean createFederateWithRti(String federationName,
String federateName,
String rtiEndpoint,
String[] federateParameters)
String getFedPerformanceParameter(String federationName,
String federateName)
Boolean terminateFederate(String federationName,
String federateName)
```

createFederate，createFederateWithRti 两种接口方法用来创建联邦进程。总的讲，前者使客户代码实现简单；后者提供给客户更大的灵活性，能为更好实现仿真目标选择可用的 RTIExec 进程，而不是通过信息服务和 RTI 服务武断地选择 RTIExec 进程。

联邦进程被成功创建后，一些相关信息应该返回给用户，以提示进程的当前状态并保存在信息服务中。接口方法 getFedPerformanceParameter 就是用来获得运行联邦的当前状态的。假如客户等待很长时间仍没得到来自服务器的结果，他可以发出请求去检查联邦的当前状态。该方法的输入参数是联邦成员名和联邦名，返回值是一个字串，用来指示联邦进程的状态。假如客户感觉联邦进程的状态不正确，他可以发出请求，终止该进程并保存当前运行参数，然后由迁移服务动态地搜寻适宜的新网格节点，启动该进程。这一方法的输入参数仍然是联邦成员名和联邦名，返回值显示这一操作是否成功。上面的论述揭示的只是每种接口方法的一般目标，每种接口的具体实现还要根据联邦服务管理的联邦的不同类型有所变化。

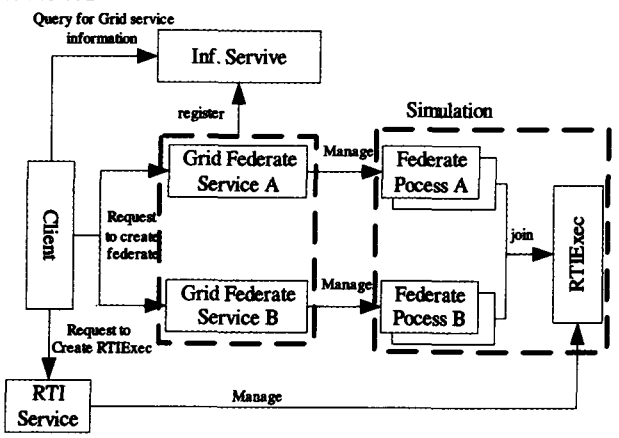

图 4 网格服务应用

图 4 显示了在本文提出的框架下进行动态交互网格服务在仿真过程中的使用情况。该框架内所包括的网格服务有信息服务、RTI 服务和联邦服务。最初，客户并不知道网格服务的地址，他可以查询信息服务上的关键词来获得该地址。正常讲，关键词是联邦临时注册的名字，也可以是它的缩写，这依靠联邦临时注册服务如何注册自己。客户获得该地址之后，与 RTI 服务联系，请求可用的 RTIExec 进程的组播地址。接下来，客户可以用获得的组播地址请求联邦服务建立一个或多个联邦进程。客户也可以对相同的联邦服务或不同的联邦服务重复这一过程，来创建多个联邦进程。在该框架下，一个联邦服务通常只管理一类联邦进程，但它能够实例化多个联邦实例。不同的实例通过联邦成员名和联邦名来区分。联邦进程实例化后，就根据组播地址加入特定的 RTIExec 进程。联邦服务负责监视联邦进程的运行状态，直到该联邦被终止或撤销。在这期间，信息服务通过动态信息动态地对联邦进程进行评估、搜寻网格节点及进程迁移操作以实现不同交互。

下面是一组客户代码。需要注意的是，当没有 RTI 服务正在运行时，查询 RTI 服务，将会返回空的字符串。这种情况下，客户应该请求 RTI 临时注册服务句柄，然后创建 RTI 服务实例。

```
rtiHandle =
inf. getRtiServiceHandleForFed(federateionName);
while(rtiHandle == null && numberofAttepts < limit)
{ritFacHandle = inf. getRtiFacHandle();
rtiFac <= resolve Rti Factory Service with rtiFacHandle;
rtiFac. createServie();
rtiHandle =
inf. getRtiServiceHandleForFed(federationName);
numberofAttempts ++;}
rti <= resolve Rti Service with rtiHandle;
endpoint = rti,. createRti();
fedFacHandle = inf. getFedFacHandle(facName,ver);
fedFac <= resolve Federate Factory Service with fedFacHandle;
fedLocator = fecdFac. createService();
fed <= resolve Federate Service with fedLocator;
fed. createFederateWithRti(federationName,
federateName,endpoint,…);
```

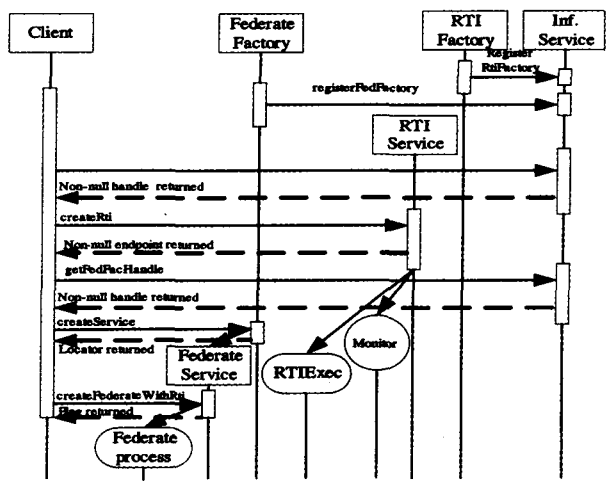

图 5 仿真过程中服务的调用

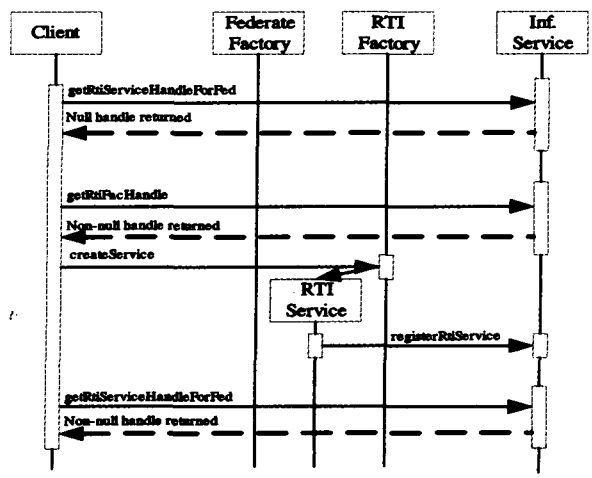

图 6 创建 RTI 服务

图5显示了RTI服务可用时的交互过程，图6则是没有RTI服务运行时客户的处理过程。当所有必须的信息被收集、资源被准备，仿真就可执行了。图7显示了当仿真正在运行时RTI服务与信息服务之间的交互，通过信息的交互系统及时对仿真运行的状态、资源及结构进行控制和调整。为了保持信息的一致性，在使用registerFedWithRtiService方法注册联邦成员时，信息服务会检查保存表。当发现该条目时，会将其删除。

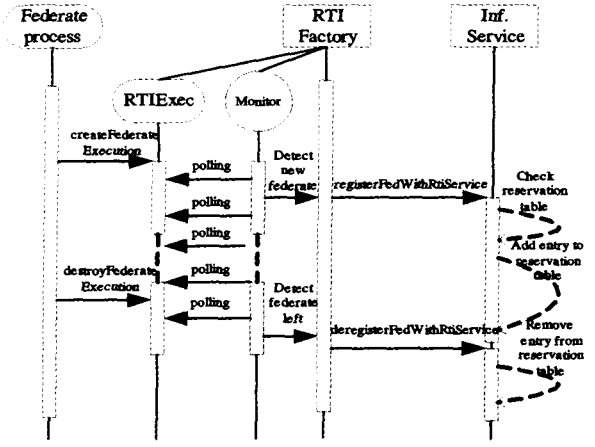

图7 RTI服务与信息服务之间的交互

5 应用实例结果

应用实例的目的是在网格环境下配置并开始仿真动态交互的时间消耗。该实验是在图1的框架的基础上进行的。

实验共检验8种情况(见表1)：(A)为获取RTI服务句柄，查询信息服务所需的时间；(B1)当RTI服务有RTIExec进程运行时，获取RTIExec组播地址的时间；(B2)没有RTIExec进程运行时，获取RTIExec组播地址的时间，该时间包括RTI服务引发RTIExec进程和为RTI服务获取组播地址的时间；(B3)没有RTI服务时，获取RTIExec进程组播地址的时间，包括获取RTI临时注册服务句柄和查询信息服务的时间、连接RTI临时注册服务并创建RTI服务实例的时间、RTI服务引发RTIExec的时间和从RTI服务获取组播地址的时间；(C)从信息服务获得联邦临时注册服务句柄的时间；(D)创建联邦进程的时间，包括连接联邦临时注册服务创建联邦服务实例的时间和连接联邦服务创建联邦进程的时间；(E)搜索新网格节点的时间，包括联邦运行状态的评估时间；(F)代码传递的时间。实验结果见表1。

表1 实验结果

Test case	Time(Seconds)
A	6.832
B1	0.150
B2	2.241
B3	3.701
C	0.141
D	0.886
E	0.123
F	0.142

应当注意在表1中的数据。情况A花费的时间大大长于其它情况，只是因为其中包含Globus设置本地运行环境的时间。B1、B2和B3是相互平行的，因为在实际仿真运行中只能有一种情况产生。因为需要引发RTIExec进程，情况B2比情况B1约长2.1s。为了发现RTIExec所需的组播地址所花费的时间就更长。利用网格服务开始一个仿真所需的总时间是：

$$A+Bx+C+D+E+F$$

其中Bx是情况B中任何一种情况。最坏的一种情况是A+B3+C+D+E+F，但总时间也没超过15s。

结论 本文提出了在网格环境下运行大规模基于HLA的分布交互仿真的动态交互框架，设计并实现了该框架下的关键网格构件、接口。在该框架下，通过动态交互实现了仿真资源的动态发现及分配、方针动态状态的监控、仿真任务及代码的动态迁移。RTIExec进程的控制过程由RTI服务来动态管理并能动态地发现；仿真模型封装在联邦服务中，以致其实现细节对用户隐藏；通过网格接口将模型组装成动态结构的分布仿真应用；不同模型可以被动态定位，系统更灵活；系统资源可以动态地配置和优化；网格的内在本质提供了系统的重用性；通信仍通过RTI。本文没对系统通信的效率进行实验，有待于下一步研究。在该框架下，系统的安全机制还没有实现，这是今后本框架完善的重点研究内容。另外，应用GRAM(Globus Resource Allocation Manager，GRAM)提高网格服务的资源管理也还是将来研究的方向。

参考文献

1 IEEE Standard 1516(HLA Rules)，1516.1(Interface Specification)and 1516.2(OMT). Sep. 2000. (http://www.dmso.mil/public/transition/hla/)

2 Kuhl F，Dahmann J S，Weatherly R. Creating Computer Simulation Systems：An Introduction to the High Level Architecture. PRT，1999

3 Dahmann J S，Morse K L. High Level Architecture for Simulation：An Update. In：Proc. of the2nd IEEE Intl. Workshop on Distributed Inter active Simulation and Real-Time Applications，1998. 32～40

4 Department of Defense，Defense Modeling and Simulation Office. RTI 1.3-Next Generation Programmer's Guide，Version 5.2002

5 Foster I，Kesselman C，The Anatomy of the Grid. International Journal of High Performance Computing Applications，2001，15(3)：200～2022

6 Globus homepage，http://www.globus.org

7 Luthi J，Grossmann S. The Resource Sharing System：Dynamic Federate Mapping for HLA-based Distributed Simulation. In：Proceedings of 15th Workshop on Parallel and Distributed Simulation，May 2001. 91～98

8 Zhang T，Zhang C，Liu Y，et al. A Design of Distributed Simulation Based on GT3 Core. In：Grid and Cooperative Computing：Second International Workshop，Shanghai，2003. 590～596

9 Morse K L，Drake D L，Brunton R P Z. Web Enabling an Rti - an XMSF Profile. Proceedings of the IEEE 2003 European Simulation Interoperability Workshop，03E-SIW-046，2003

10 Cai W，Turner S J，Zhao H. A Load Management System for Running HLA-based Distributed Simulations over the Grid. In：Proceedings of the 6th IEEE International Symposium on Distributed Simulation and Real Time Application，Oct. 2002. 7～14

11 Zajac K，Bubak M，Malawski M，et al. Towards a Grid Management Systems for HLA-based Interactive Simulations. In：Proceedings of the 7th IEEE International Symposium on Distributed Simulation and Real-Time Applications，Oct. 2003. 4～11

12 Zajac K，Tirado-Ramos A，Zhao Z，et al. Grid Services for HLA-based Distributed Simulation Frameworks. In：1st European Across Grids Conference，Spain，Feb. 2003. 147～154

13 Zong Wenbo，Wang Yong，Cai Wentong，et al. Grid Services and Service Discovery for HLA-Based Distributed Simulation. Proceedings of the Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications(DS-RT'04)，2004