

自定义数据上传应用在 OPNET 中的实现

李 云^{1,2}, 严海蓉³, 王玉林¹, 邹德财^{1,2}

LI Yun^{1,2}, YAN Hai-rong³, WANG Yu-lin¹, ZOU De-cai^{1,2}

1.中国科学院 国家授时中心, 陕西 临潼 710600

2.中国科学院 研究生院, 北京 100039

3.西安工业大学 计算机学院, 西安 710032

1.National Time Service Center, the Chinese Academy of Sciences, Lintong, Shaanxi 710600, China

2.The Graduate School of the Chinese Academy of Sciences, Beijing 100039, China

3.Department of Computer Science, Xi'an Technological University, Xi'an 710032, China

E-mail: liyun04@mails.gucas.ac.cn

LI Yun, YAN Hai-rong, WANG Yu-lin, et al. Realization of custom application of uploading data in OPNET. *Computer Engineering and Applications*, 2007, 43(20): 113-116.

Abstract: The custom application of uploading data is added in the application layer, as a standard application, to simplify the complex process of parameter configuration in the custom application and profile, and to complete the special application. The monitoring and controlling network system is simulated using the new application, and the simulation results show that the standard application can cooperate with the added. Compared with the custom application and profile, it is more simple and embodying the characteristic of the special application.

Key words: OPNET; process model; application attribute; simulation

摘 要: 为了简化网络仿真中自定义应用和业务中繁琐的参数配置过程, 完成数据上传的特殊应用, 在 OPNET 中实现了将自定义数据上传应用添加到应用层作为一标准应用。利用自定义应用对一网络监控系统进行了仿真, 运行过程正常, 仿真结果表明标准应用和添加的这一应用可以协同工作。与自定义应用和业务相比, 它能简化参数配置的复杂过程并体现特殊应用的特点。

关键词: OPNET; 进程模型; 应用属性; 仿真

文章编号: 1002-8331(2007)20-0113-04 文献标识码: A 中图分类号: TP393

OPNET 应用层标准模块已经提供多种应用协议, 包括数据库(Database)、电子邮件(E-mail)、文件传输(Ftp)、超文本传输协议(Http)、打印(Print)、远程登录(Remote Login)、视频会议(Video Conferencing)、音频(Voice)。如果仿真的业务不在 OPNET 提供的标准应用协议之内, 这时可以采用自定义应用和业务。将业务首先分解为若干个任务, 然后通过精确定义应用的每个动作来逼近实际的应用行为; 或者将仿真业务作为一种自定义的应用协议加入 OPNET 标准应用层模块, 这要涉及到修改 OPNET 应用层的标准进程模型和相关的头文件。本文关注后者的实现过程, 下面将以自定义数据上传应用协议为例详细说明其实现过程。

1 自定义数据上传协议的介绍

本文定义的数据上传协议为: 客户端每隔一定的时间向服务器发送数据文件, 服务器收到文件后给客户端一个响应, 表示收到了客户端发送的文件。绝大部分数据从客户流向服务器, 服务器将这些数据储存起来。同 OPNET 中的标准应用一样, 数据上传协议也有两层体系结构, 分别为客户提出请求和

服务器收到请求返回应答^[1]。

上传应用的属性有: 客户发起上传工作的时间间隔(Upload Interarrival Time), 上传文件的平均大小(File Size), 上传服务器的符号名称(Symbolic Server Name), 以及服务质量参数(Type of Service), 用于对应用中优先级的规定。此业务主要运用在监控系统中, 各个客户端以随机或固定时间间隔的方式向服务器端上传监控数据。监控系统对实时性的要求比较高, 因此传输层采用 UDP 协议。

2 应用层模块的内部结构分析

在 OPNET 中, 应用层模块主要包括四级进程模型, 如表 1 所示, 从上到下依次为父进程和子进程的关系。

表 1 应用层模型

| 模块 | 进程名 |
|------------|-----------------|
| NAM 网络管理模型 | gna_clsvr_mgr |
| PM 业务管理模型 | gna_profile_mgr |
| AM 应用管理模型 | gna_应用_mgr |
| AC 客户端模型 | gna_应用_cli |

NAM(Network Application Manager)是网络应用层管理模块,即应用层的根进程 gna_clsivr_mgr.pr.m,它主要负责向目的地发送业务连接包,如果业务会话成功建立,则生成与该业务相对应的业务规格的流量,当业务发送完毕时关闭会话^[2]。在进程 gna_clsivr_mgr.pr.m 的状态 spawn profile 中通过函数 gna_profile_spawn 为中断代表的业务启动一个业务管理器 PM(Profile Manager),即创建一个子进程 gna_profile_mgr.pr.m。

PM(Profile Manager)是业务管理模块,该模块为应用层的应用建立一个子进程,从而并行的生成隶属于不同应用类型的业务。在进程 gna_profile_mgr.pr.m 中,spawn 状态为不同的应用产生应用管理器 AM(Application Manager),创建进程 gna_email_mgr, gna_ftp_mgr 等。在 SPAWN 状态的入口程序中具体实现如下:

```
...
case GnaT_ApType_Print:
{...
application_mgr_prohandle_ptr[ith_app]=op_pro_create("gna_print_mgr",OPC_NIL);
...
}
...
```

AM(Application Manager)是应用管理模块。一个应用服务器有多个客户端,AM 将为不同的客户端分别创建客户端子进程,即 AC(Application Client)客户应用模块,进程 gna_email_cli, gna_print_cli 等。以打印应用为例,用到的数据结构是 GnaT_Print_Desc,其中各个分量是打印应用的具体属性,对应于(Print)table 对话框中的相关属性。进程 gna_print_mgr,在状态 send file to printer 中通过函数代码 print_cli_phandle=op_pro_create("gna_print_cli", OPC_NIL)实现客户端子进程的创建。

前面的高三层模块构建了应用层建模的框架,而 AC(Application Client)客户应用模块是整个应用层模型最底层的进程。应用协议的实现主要在 AC,同时各种应用协议的区别也主要体现在该模块中。以打印应用为例,用到的数据结构是 GnaT_Cli_Print_Params_Info,通过分量 app_info_ptr 和上面三层进程建立联系。

并非八种标准应用都遵循以上建模结构,视频会议(Video Conferencing)、音频(Voice)没有客户端和服务端,而是主动方 calling 进程和被动方 called 进程。

3 协议的实现

通过以上对应用层模块内部结构的分析,要添加一个标准应用,关键是要创建应用管理进程和客户进程,即应用层模块中处于最底端的两层。在此基础上,在网络应用层管理模块和业务管理模块,即根进程 gna_clsivr_mgr.pr.m 和 gna_profile_mgr.pr.m 中添加相应的代码。同时,对程序中用到的数据类型、变量和常量,在头文件中要对其进行声明和定义。这里先介绍协议的实现过程中所用到的数据结构在头文件中的定义,然后以从父进程到子进程的顺序对协议的实现进行详细说明。

3.1 头文件中添加相应的数据结构、变量与常量

3.1.1 对头文件 gna_clsivr.h 的修改

在枚举类型 GnaT_ApType 中,添加一分量 GnaT_ApType_Upload,用于表明数据上传的应用类型。

为下文要创建的进程 gna_upload_mgr 和 gna_upload_cli 分

别定义结构类型 GnaT_Upload_Desc 和 GnaT_Cli_Upload_Params_Info。GnaT_Upload_Desc 用于对上传应用(Upload)属性的描述,包括文件上传时间间隔、上传文件的大小、服务器的符号名字等。GnaT_Cli_Upload_Params_Info 包含应用必须的参数,在每一次会话的开始,由应用管理器传给应用客户端。

3.1.2 对头文件 gna_api.h 的修改

在应用名字 GnaT_Application_Name、应用类型 GnaT_Application_Type 和应用端口 GnaT_App 等枚举类型中分别添加上传应用相关的分量 GnaC_App_Upload、GnaC_App_Type_Upload 和 Upload。GnaT_Application_Type 枚举类型变量广泛应用于根据应用类型调用应用的解析函数、创建应用管理进程等的函数中。在程序中,根据 GnaT_App 型变量获取相应的应用类型,根据 GnaT_Application_Name 型变量获取包括传输协议的应用各种属性。

此外,增加宏定义:#define GNAC_APP_UPLOAD"Upload"。

3.2 修改网络应用管理进程模型(gna_clsivr_mgr)

gna_clsivr_mgr 是整个应用层的根进程,用于建立业务会话、生成相应的业务流量,在此进程的以下各个函数中分别添加与上传应用相关的程序代码。

在函数 Gna_clsivr_mgr_conf_read()中添加程序代码,以获取上传应用的传输协议,服务状态以及对一些状态变量进行初始化。

函数 Gna_client_profile_parser()的功能是客户端的业务解析,把一些符号名用网络中实际的名字来替代。对每一个应用都创建一个结构来存储和应用相关的业务,并将这个结构类型的变量传给应用管理进程模型。在此函数中添加代码以实现与上传应用相关的业务的解析,同时,定义一个 GnaT_Upload_Desc* 类型的变量 upload_appl_ptr=OPC_NIL 以满足所添加的代码段的需要。在状态变量 SV 中定义一个整形变量 upload_max_stat_index,用于表示上传应用中统计量的最大个数。函数 gna_clsivr_mgr_port_to_apptype()根据应用注册的端口返回应用类型。在此添加代码实现与上传应用相关的功能。

由于在应用层添加了一个新的应用,在进程 gna_clsivr_mgr 的头文件块 HB 中,将宏 NASC_NUM_APPS 的值由原先的 12 修改为 13。如果保留原值,在程序的运行中提示错误如下:

```
Simulation terminated by process ( gna_clsivr_mgr ) at module
(top.Office Network.upload_server.application)
```

```
T(109.029),EV(933),MOD(top.OfficeNetwork.upload_server.application),PROC(op_sim_end)
```

```
Error in GNA client-server application manager(gna_clsivr_mgr):
Invalid service type carrieed by received packet
```

3.3 业务管理进程模型中增加上传应用启动程序

进程 gna_profile_mgr.pr.m 是根进程 gna_clsivr_mgr 的一级子进程,其 spawn 状态针对应用类型的不同而创建应用管理进程,启动新的应用,同时从高层析取合理的数据结构并将它传递给应用管理器。在此增加代码以创建上传应用 Upload 管理进程。

3.4 在应用进程配置模型中增加应用属性和上传应用分析程序

三个进程模型 Task_configuration, Application_configuration 和 Profile_configuration 分别对任务定义、应用定义、业务配置界面的参数进行解析,存储到相应的变量中,等待应用层进程

模型的调用^[3]。其中应用配置进程模型 Application_configuration 主要设置应用层各种应用的属性并对其进行分析, 既然添加了自定义的上传协议在应用层, 就要在应用配置进程中设置其属性, 并添加应用解析函数。

在进程 application_config.pr.m 中, 通过 Model Attributes 命令, 增加 Upload 应用及其相关属性。在函数块 FB 中增加函数 gna_upload_desc_parser, 对 Upload 应用的具体属性进行解析, 将 (Upload)Table 对话框中的各个属性逐一赋值给 GnaT_Upload_Desc* 类型的变量 upload_desc_ptr 的分量。头文件块 HB 中要对此函数进行说明。

在函数块 FB 中函数 gna_application_desc_parser() 的主要功能是调用具体应用的解析, 返回应用具体描述结构变量, 因此要添加相应代码解析 Upload 应用。最终上传应用的属性对话框如图 1 所示。

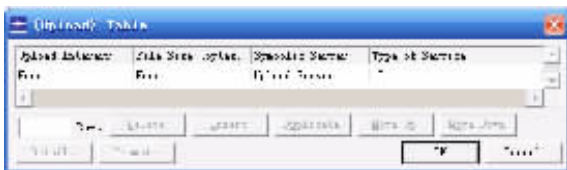


图 1 上传应用的属性

3.5 创建上传 Upload 应用管理进程和客户进程

3.5.1 建立 Upload 应用管理进程 gna_upload_mgr

上传 Upload 应用管理进程主要功能是为不同的上传应用客户建立客户进程 gna_upload_cli。进程模型如图 2 所示。

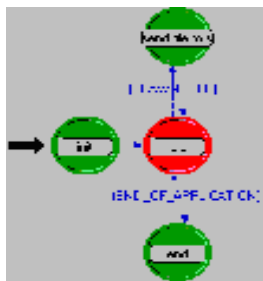


图 2 上传应用管理进程模型

init 状态主要为客户进程 gna_upload_cli 中用到的指向 GnaT_Cli_Upload_Params_Info 型的指针变量赋值, 安排了两个自中断, 为进程从 idle 状态进入 send file to server 状态和进入 end 状态创造条件。在 idle 状态中要读取中断类型和中断代码。

send file to server 状态创建客户进程 gna_upload_cli, 将数据传送给服务器, 并且根据本次上传的结束时间, 安排下一

次文件上传。每一次文件上传都要建立一个新的客户进程。

end 状态释放所用变量占据的内存空间, 销毁进程结束上传应用。

3.5.2 建立上传 upload 客户进程 gna_upload_cli

客户进程 gna_upload_cli 和传输适应层 TPAL 建立连接, 然后上传文件。进程模型如图 3 所示。

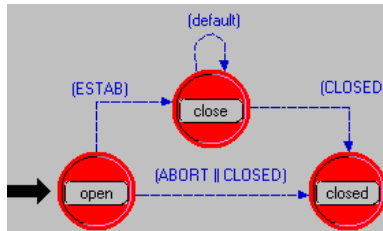


图 3 上传应用 upload 客户进程模型

Open 状态的入口程序代码与 TPAL 层建立连接, 通过函数 tpal_app_session_open, 建立和传输适应层的连接, 同时返回应用层和传输适应层公用的接口信息 ICI。对全局和局部统计量进行注册。和传输层建立连接后, 进行文件的上传, 改写统计量的值, 这部分功能在 Open 状态的出口程序中完成。

Close 状态等待服务器发来的接收确认消息。在 Closed 状态关闭 TPAL 连接, 释放占用的内存空间, 销毁进程。

3.5.3 声明子进程

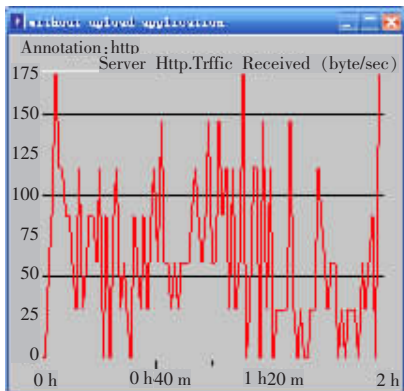
声明进程 gna_upload_cli 是进程 gna_upload_mgr 的子进程, 进程 gna_upload_mgr 是进程 gna_profile_mgr 的子进程。

4 仿真运行

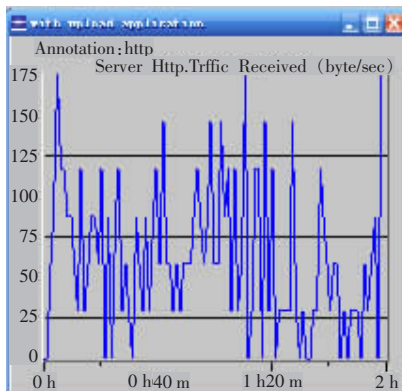
为了检验添加的这一应用是否影响 OPNET 模型库中自带的标准应用的运行, 对自带的任一项目 Application_Configuration_Example 在 standard_applications 场景下进行仿真。从结果可以得到标准应用电子邮件 (E-mail)、文件传输 (Ftp)、超文本传输协议 (Http) 的客户端响应时间以及各节点的发送和接收流量。HTTP 服务器的接收流量在添加上传应用前后的仿真结果如图 4, 由图得出添加应用前后仿真结果没有变化。

又采用了一网络监控系统作为仿真对象, 该系统中有一 10 个客户端 1 个服务器, 拓扑结构如图 5 所示。网络的主要功能是数据上传业务。客户端实时从终端设备收集数据, 而后间隔一定的时间向服务器端发送数据文件。

由于系统的需要, 要求客户端从终端设备收集两倍多的数据发送给服务器。为了分析系统性能会发生怎样的变化, 需要收集统计量服务器的负载和系统的延时来加以验证。有两倍的



(a) 添加上传应用前



(b) 添加上传应用后

图 4 添加应用前后的仿真结果

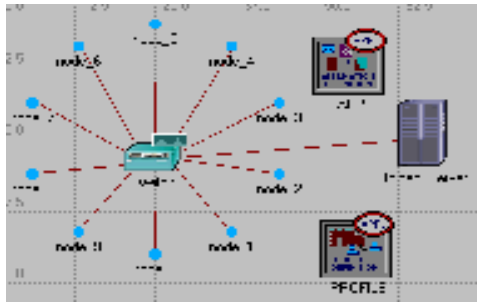


图5 监控系统拓扑结构

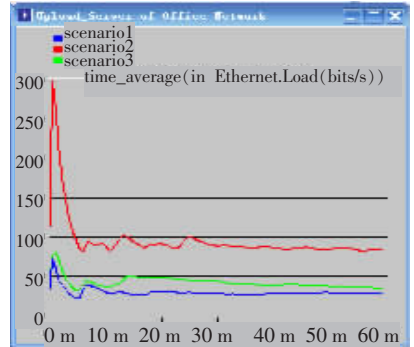
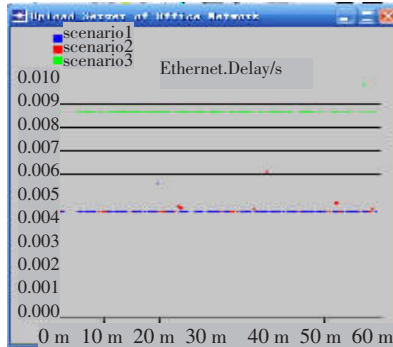


图6 仿真结果比较

数据从客户端流向服务器,在应用配置中表现为两种形式,一种发送文件的大小不变,上传间隔减半,另一种上传间隔不变,发送文件的大小加倍。

配置三个场景,scenario1 表示原始配置,scenario2 表示上传时间间隔减小的配置,scenario3 表示传输文件大小加倍的配置。仿真运行正常,确认自定义上传应用能不受标准应用的影响可以独立的运行。结果如图 6 所示,从图可以看出:文件大小加倍网络延时几乎加倍,对服务器负载的影响不大。上传时间间隔减半网络延时几乎不变,对服务器负载的影响加倍。

如果采用引言中提及的定义应用和业务的方法对其进行仿真,要进行任务的定义、应用的配置、业务的配置以及符号名和实际名字的一一映射,这是一个比较繁琐的过程。现在将其添加到应用层作为标准应用后,只进行简单的应用配置就可以实现仿真。

5 结论

通过分析应用层节点模型的内部结构,创建底层进程,向根进程及部分子进程添加代码,将自定义上传应用添加到 OPNET 的应用层,满足了特殊应用的需求,简化了采用自定义应用和业务配置的复杂过程。从后面的仿真分析看,添加的应用和标准应用可以相互独立正常的工作,表明将自定义上传应用添加到 OPNET 的应用层这一方法的可行性,及其过程的正确性。(收稿日期:2006 年 11 月)

参考文献:

- [1] 王文博,张金文.OPNET Modeler 与网络仿真[M].北京:人民邮电出版社,2003:308.
- [2] 陈敏.OPNET 网络仿真[M].北京:清华大学出版社,2004:300.
- [3] Application Model User Guide.OPNET 10.0 Product Documentation Inc,2003.

(上接 64 页)

大改进,图 5 是对两个算法在时间上的比较,简化的模型是前述实验中采用的 bunny 模型。



图3 QEM 简化结果的线框表示



图4 FMS 简化结果的线框表示

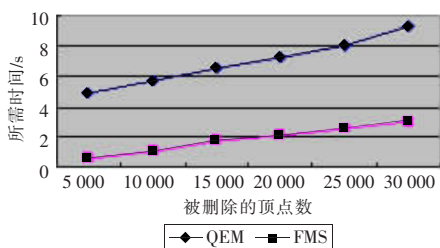


图5 QEM 与 FMS 的效率比较

通过图 5 可以看到 FMS 在时间效率上比 QEM 表现要好,并且随着简化的顶点越多,效率提高越明显,这样在刑侦现场重现或者虚拟会议等实时系统中就可以迅速地得到简化后的模型,提高模型绘制的速度。

5 结论

为了能在类似于刑侦现场重现、虚拟会议等实时系统中更快的简化网格模型,本文提出了一种快速的网格模型简化算法 FMS,它属于边收缩型的算法。它通过顶点权值来描述一个顶点的重要性,并通过顶点权值得到边的收缩代价,与 QEM 采用的顶点误差相比,FMS 采用的顶点权值的计算更为简单。该算法还利用在网格模型中数量最少的图形元素顶点的优先权队列来控制边的收缩顺序,提高了优先权队列更新的效率及模型简化的速度。(收稿日期:2006 年 11 月)

参考文献:

- [1] Garland M,Heckbert P S.Surface simplification using quadric error metrics [C]//SIGGRAPH.Proceedings of SIGGRAPH 97,International Conference on Computer Graphics and Interactive Techniques 1997.New York USA:ACM Press,1997:209-216.
- [2] Garland M,Zhou Yuan.Quadric-based simplification in any dimension[J].ACM Transaction on Graphics,2005,24(2):209-239.
- [3] Vieira A W,Velho L,Lopes H,et al.Fast stellar mesh simplification[C]//SIBGRAPI.XVI Brazilian Symposium on Computer Graphics and Image Processing,Brazil,2003:27-34.
- [4] Lee Chang-ha,Varshney A,Jacobs D W.Mesh saliency [J].ACM Transactions on Graphics ACM,2005,24(3):659-666.
- [5] Birlhelmer H,Soetebier I,Sahm J.Efficient representation of triangle meshes for simultaneous modification and rendering[J].Lecture Notes in Computer Science,2003:925-934.