

基于 Creator/Vega Prime 的单件生产 系统虚拟现实仿真

李明, 汪 峥

(东南大学 自动化学院, 江苏 南京 210096)

摘要: 为了对一个单件生产系统调度过程进行虚拟现实仿真, 文中首先采用多线程方式编写了单件生产系统的实时仿真程序, 此程序可仿真各工件在各机器上的调度过程。为了使仿真过程可视化, 使用三维建模工具 Creator 建立了这个单件生产系统的模型, 并在 VS2003 环境下由仿真程序调用 Vega Prime 工具包对模型进行驱动, 最终实现了对单件生产系统调度过程的虚拟现实仿真。从仿真结果来看, 可以清晰直观地看到该单件生产系统的工作过程, 证明此方法正确、有效, 并可以推广到更复杂的生产系统中。

关键词: Creator; Vega Prime; 单件生产系统; 虚拟现实仿真

中图分类号: TP391.9

文献标识码: A

文章编号: 1673-629X(2011)01-0197-05

Virtual Reality Simulation of One-of-a-Kind Production System Based on Creator and Vega Prime

LI Ming, WANG Zheng

(School of Automation, Southeast University, Nanjing 210096, China)

Abstract: In this paper, a software is developed by using the multi-thread method to fulfill the real time simulation programme of a one-of-a-kind production system. In order to visualize the simulation, a model of the system is created by Creator and driven to fulfill the virtual reality simulation of the production system by using the Vega Prime toolkit in VS2003. The working process of the one-of-a-kind production system can be seen clearly in the virtual reality environment. The simulation result indicates that this method is feasible.

Key words: Creator; Vega Prime; one-of-a-kind production system; virtual reality simulation

0 引言

虚拟现实技术是客观世界中的事物在计算机上的本质实现。虚拟现实技术的核心是建模和仿真, 它在国内外被广泛应用。典型的应用领域有^[1-3]: 军事领域、模拟训练、城市规划、房地产开发、古迹复原等。文献[4]基于“Creator+Vega+VC”视景仿真方法, 实现了巡逻攻击导弹作战仿真系统。文献[5]建立了挖掘机的三维模型, 并完成了挖掘工艺流程的动态仿真。文献[6]开发了船舶机舱虚拟现实仿真系统, 建立了逼真的船舶机舱虚拟场景, 实现了人在虚拟机舱环境中的漫游。文献[7]讨论了基于 Creator 和 Vega 的视景仿真技术在交通仿真中的应用。文献[8]介绍了矿井

视景仿真的实现过程。

文中将视景仿真方法引入到机械加工领域, 对一个单件生产系统的工作过程进行虚拟现实仿真。所谓单件生产, 即是每个产品都被看作与其他产品不同, 因此, 可认为此种生产方式下产品种类数很多, 而每种产品的数量只有一个。

1 利用 Creator 建模

1.1 单件生产系统模型介绍

MultiGen-paradigm 公司提供的三维建模系列工具 MultiGen Creator, 拥有对实时应用优化的 Open-Flight 数据格式, 强大的多边形建模、矢量建模、大面积地形精确生成功能, 以及多种专业选项及插件, 能高效、最优化地生成实时数据库, 并与后续的实时仿真软件紧密结合, 在视景仿真、拟训练、城市仿真、交互式游戏及工程应用、科学可视化等实时仿真领域有着广泛的应用^[9,10]。

鉴于上述优点, 文中选用 Creator 作为建模工具,

收稿日期: 2010-04-28; 修回日期: 2010-07-03

基金项目: 国家自然科学基金项目(60974096)

作者简介: 李明(1981-), 男, 安徽利辛人, 硕士研究生, 研究方向为计算机集成制造; 汪 峥, 副教授, 研究方向为制造系统控制与设计、离散动态系统等。

建立单件生产系统的模型,如图 1 所示。

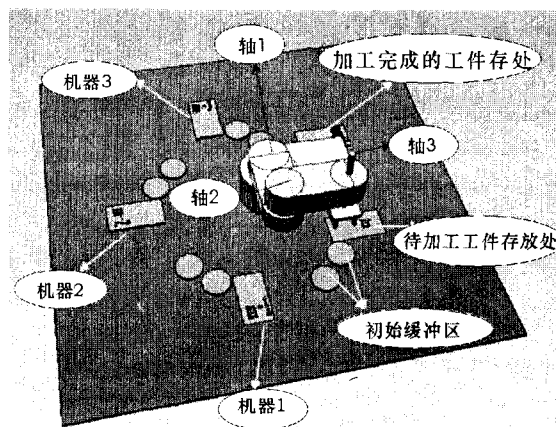


图 1 单件生产系统模型

位于模型中间位置的是一个有 4 个自由度的机械臂,它负责将工件搬运至机器上加工或放至缓冲区。轴 1 和轴 2 都有一个转动的自由度,轴 3 有 2 个自由度,可以做转动和上下运动。系统中有 3 台机器,每个机器旁有个容量为 2 的缓冲区。由于笔者不关心在机器上的具体加工场景,故用一个位置代表机器,将工件放在此位置上代表在这个机器上加工。

由 Creator 建立的模型为 .flt 文件,机械臂模型为 robotArm.flt。此类文件可以由 Lynx Prime 图形界面直接加载到视景里,也可以在程序里作为 setFileName() 函数的参数将此模型赋予某 vpObject 对象。

1.2 机械臂的节点层级视图

因为机械臂有 4 个自由度,而且抓具要打开和闭合,所以要为其相应节点添加动态特性,这就需要用到 DOF(自由度)技术。DOF 是一个可为其子节点所表示的图形添加动态特性的节点类型。一个 DOF 创建一个本地坐标系,它所控制的图形可绕其中的坐标轴进行旋转或沿指定轨迹进行移动。机械臂的节点层级视图见图 2。图中 db 为根节点,gl 为机械臂组节点,g2 为轴 1 以下的台座部分,axis1,axis2,axis34 分别是轴 1、轴 2 和轴 3 所对应的 DOF 节点。zhuazi1 和 zhuazi2 分别是抓具的两个爪子对应的 DOF 节点。

2 利用 Vega Prime 驱动模型

2.1 在 Lynx Prime 下进行预定义

Lynx Prime(LP)是一种 GUI 工具,为用户提供了一个简单的开发界面,可根据仿真需要快速开发出合乎要求的视景仿真应用程序。它具有向导功能,能对 Vega Prime 的应用程序进行快速创建、修改和配置^[11]。

可以利用 LP 将所需要的模型配置在适当的位置,可以设置模型的运动方式,设置视点及其运动方式,设置模型用于碰撞检测的相交矢量等等。所有在 Lynx Prime 中对各种面板参数的定义都可以在编程中直接进行,但这样没有在界面上预定义来得简单和直观,会增加编程量,并且通过在界面上的预定义参数可以立即通过动态预览看到效果,以便及时修改参数,直到满意为止,节约了应用系统的开发时间和成本,这时把各种配置参数的定义保存在一个文件(*.acf)中。

2.2 动态生成工件

利用 LP,可以把整个生产系统加载到视景中,并可以调整到较好的观察角度。虽然利用 LP 非常方便,但是有些情况下还是需要在程序中进行动态生成。比如说需要生成一批工件,在 LP 中需要逐个加载,而且要事先知道数量,而用动态加载就可以根据实际需要进行生成。以下代码为动态生成工件的 vpObject 类对象,并放置到视景的适当位置。例如:

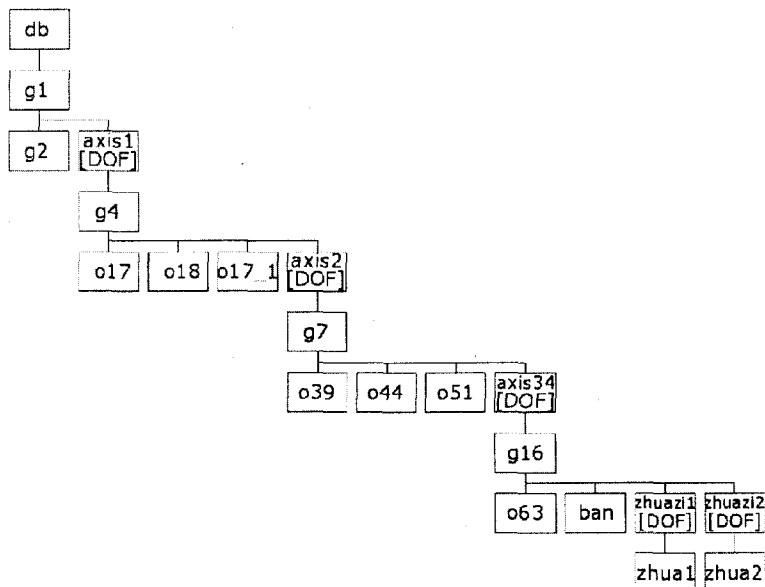


图 2 机械臂的节点层级视图

```
vpObject * work = new vpObject(); // 创建一个 vpObject 对象
vpScene * scene = * vpScene::begin(); // 找到场景节点
work->setPosition(3, -143, -100); // 设定该对象的位置
work->setFileName("work2.flt"); // 将工件的模型赋予该对象
scene->addChild(work); // 将该对象在场景里显示
```

2.3 Vega Prime 应用程序框架结构

Vega Prime 主程序流程包括初始化、定义、配置、帧循环、关闭五个部分^[12]。

- (1) 初始化: 实现变量初始化和内存分配。
- (2) 定义: 调用一个以 acf 文件名为参数的 define() 函数, 表示将哪个 acf 文件加入程序。
- (3) 配置: 实现场景内对象的初始化配置, 如机械臂的各个轴、爪、零件等对象的获取。

(4) 帧循环:以 `beginFrame()` 开始, `endFrame()` 结束, 构成整个应用程序的刷新与循环。程序的主要工作在帧循环里进行。

(5) 关闭:清除场景中的对象, 内存释放, 结束整个程序。

文中的程序主框架结构如下:

```
void main(int argc, char * argv[])
{
    vp::initialize(argc, argv); //初始化 vega prime
    myApp * app = new myApp; //自定义 myApp
    if(argc <= 1)
        app->define("jixiebi1.acf"); //加载 acf 文件
    else
        app->define(argv[1]);
    app->configure(); //配置
    app->run(); //进入帧循环
    app->unref(); //清除场景中的对象并结束仿真
    app->shutdown(); //关闭系统
}
```

2.4 机械臂关节运动的实现

`vpApp` 类用来定义一个典型的 VP 应用的框架。它在 `vpApp.h` 中定义。所有子方法 (member method) 都被内嵌了。使用者可以拷贝和修改 `vpApp` 类。`vpApp` 的主体封装了 VP 应用中经常用到的 `vpKernel` 的功能。`vpApp` 类控制实时功能 (包括定义 ACF、配置仿真类、仿真循环、更新和退出)。文中从 `vpApp` 类中派生出 `myApp` 类。在该类中, 设置指针变量指向各 DOF 节点, 如 `vsDOF * dof`。通过函数 `find_named()` 找到机械臂各 DOF 节点, 并让类中的成员变量指向对应的 DOF 节点。例如:

```
vpObject * myClam;
myClam = vpObject::find("robotArm"); //找到机械臂
vsNode * tempNode;
vsDOF * dof2;
tempNode = myClam->find_named("axis2");
if(tempNode->isExactClassType(
    vsDOF::getStaticClassType()))
//判断找到的节点类型
dof2 = (vsDOF *)tempNode; //指针类型转换
```

上述程序段的意思为: 定义一个 `vpObject` 型的指针 `myClam`, 然后利用 `find()` 方法找到机械臂的模型, 并让 `myClam` 指向这个模型; `find_named()` 可按名字找到任意类型的节点, 返回值是个 `vsNode` 型指针, 所以定义一个 `vsNode` 型指针和一个 `vsDOF` 型指针, `vsNode` 型指针用来接收 `find_named()` 的返回值, 判断一下找到的这个节点确实是我们需要的 `vsDOF` 节点后, 将指针类型做强制转换, 这样以来, 就可以得到一个 `vsDOF` 类型的指针变量, 指向机械臂的轴, 通过对这个 DOF

的操作可以达到使机械臂运动的目的。例如:

```
dof2->setRotateH(1, true); //使轴 2 旋转 1 度
```

2.5 抓具抓起工件的实现

抓具即为机械臂最末端的爪子。机械臂通过抓具将工件抓起并搬运至机器或缓冲区。实现的方法为: 当抓具闭合时, 通过函数 `addChild()` 将工件加载为抓具的子节点, 工件就会随抓具一起移动。当工件被运送至需要的位置时, 抓具打开, 通过函数 `removeChild()` 把工件从抓具节点上移除, 工件就不会再随抓具移动。

这样一来, 就可以使得机械臂通过各关节的运动到达工件的位置, 抓起工件, 然后放到需要的位置。

2.6 机械臂运动速度的控制

文中是通过各 DOF 节点来实现机械臂的动态特性, 而 DOF 节点没有直接的速度控制函数, 所以通过控制帧的刷新频率和设置每帧里机械臂的动作幅度两方面来完成对机械臂运动速度的控制。每一帧里仿真程序均以运行时仿真状态参数为输入, 从根节点开始依次遍历各子节点, 用相关联的驱动模型驱动各个节点, 更新各节点的状态信息, 完成本次循环。所以帧循环刷新的频率和视景的规模有关, 文中所涉及的系统规模较小, 帧刷新频率很快, 所以很容易出现机械臂瞬间转到指定位置的情况。所以通过自定义函数 `renew()` 来控制刷新率。

```
void myApp::renew()
{
    Sleep(30);
    endFrame();
    update();
}
```

由代码可知, 在每帧的动作结束后, 休眠 30 毫秒, 再结束帧, 开启新的帧。这样, 帧循环的频率被控制在 33/s 左右, 既不影响画面的连续性, 又可以针对此频率设置在一帧里机械臂合理的动作幅度, 使得最终机械臂按照需要的速度运动。

3 单件生产调度仿真的实现

3.1 工件类的设计

单件生产系统里每个工件的工序和每道工序所需要的时间可能都是不一样的。另外, 在仿真过程中需要知道每个工件的状态, 如: 工件的位置、是否在加工、下道工序在哪台机器等。所以设计工件类, 将工件的各种信息如工序、每道工序加工时间等作为类的成员变量。以下为工件类中部分变量和函数:

```
void StartThread(); //为工件创建线程
static unsigned _stdcall PartProc(LPVOID lpParam); //线程函数
int m_part_ID; //工件号
```

```

int m_part_buffer; //所在缓冲区
int m_in_machine; //所在机器
int m_process_time[MACHINE_NUM]; //加工时间
int m_order[MACHINE_NUM]; //工序
int m_part_complete; //加工结束标志
int m_process_ed; //已加工工序数
HANDLE m_hPart; //线程句柄
vpObject * m_work; //此工件对应的 vpObject 类对象

```

由于可能有几个工件同时被加工,那么就需要给这几个工件的加工分别计时。为了使逻辑上更加清晰,相关变量的耦合度尽可能小,在工件类中设置一个创建线程函数,这个函数会给该工件对象创建了一个线程,创建后处于挂起状态,当工件进入到初始缓冲区时,认为工件已经进入了系统,于是将此工件的线程开启,当工件所有的工序都结束时,将此工件的线程关闭。每个在生产系统中的工件都对应着一个线程,在这个线程中模拟被加工的过程,在文中就是使得工件在一道工序的加工过程中,将其状态保持为忙,而在这道工序结束后,将其标志改为空闲。工件线程的工作流程如图 3 所示。

工件线程的工作过程可以从控制台方式下显示的信息中看出(见图 6)。例如:工件 11 在 11:26:26 开始在机器 2 上加工,本道工序需要 45 秒,11:13:11 时,也就是 45 秒后,程序得知此道工序已经加工完成,将

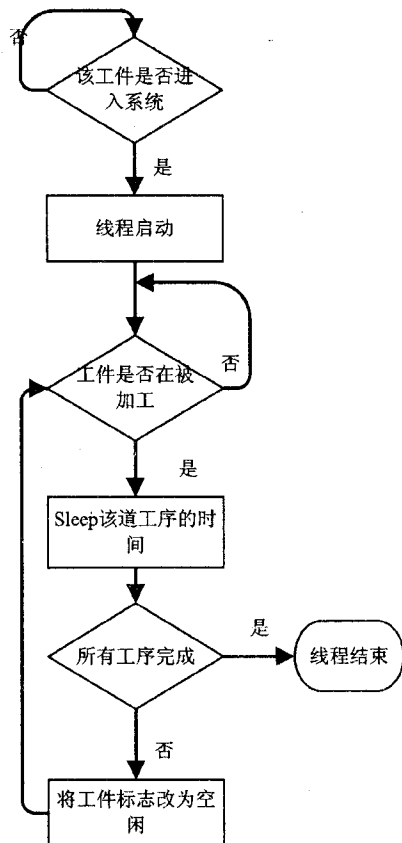


图 3 工件线程的流程

零件的状态改为空闲,可以对此工件进行下一步的操作了。

在工件类中还有个特殊的成员变量 `vpObject * m_work`,它是一个指向 `vpObject` 类对象的指针。通过这个变量,工件类的对象和 Vega Prime 中的虚拟现实场景里的工件便可以一一对应起来,工件类对象状态的改变可以直接在场景里显示出来,达到了仿真过程可视化的目的。

3.2 机械臂的调度逻辑

机械臂的调度逻辑如图 4 所示。

在此单件生产系统中,机械臂会挑选优先级最高的工件进行加工,但前提是这个工件所需要的机器是处于空闲状态的,如果这个前提不满足,则会加工优先级次高的工件。在文中,工件的优先级是由工件号决定的,工件号越小则优先级越高。如果使用其他的调度策略,只需根据调度策略计算出各工件的优先级即可,而调度逻辑不需要变化。

4 仿真结果及结论

仿真开始后,可以看到机械臂将按照各工件的优先级和它们的工序依次将工件抓取至各机器加工,加工完毕的工件被放至机器的缓冲区等待下道工序,所有工序都结束的工件被送离系统。同时,在控制台实

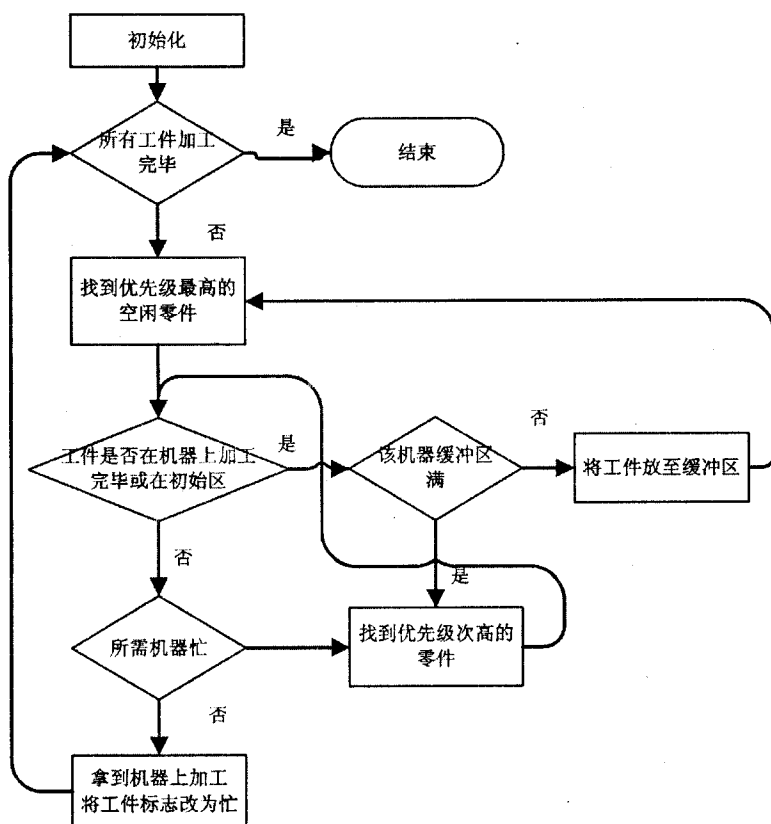


图 4 机械臂调度逻辑

时输出每个事件及其发生的时间。通过仿真画面和控制台的信息,可以完全掌握整个仿真过程的信息。图5为仿真画面的截图。图6为部分控制台显示的信息。

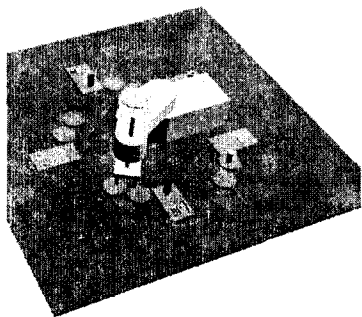


图5 仿真画面截图

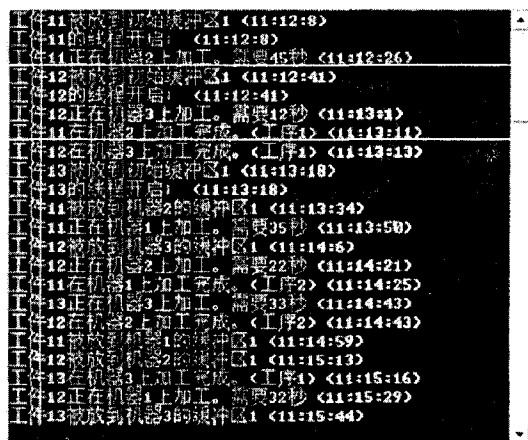


图6 控制台方式下显示的部分信息

文中基于多线程技术对一个简单的单件生产系统的工作过程进行了实时仿真,在此基础上,使用 Creator 对这个单件生产系统建模,并用 Vega Prime 对此模型

进行驱动,最终实现了关于这个单件生产系统的虚拟现实仿真。从仿真结果来看,可以清晰直观地看到单件生产系统的工作过程,证明此方法正确、有效。

参考文献:

- [1] 韦有双,杨湘龙,王飞. 虚拟现实与系统仿真[M]. 北京:国防工业出版社,2004.
- [2] 胡忠东,樊爱华. 虚拟现实工具[J]. 计算机仿真,1997, 14(5): 61-64.
- [3] 张秀山,徐荣花,胡庆丰,等. 虚拟现实技术及编程技巧[M]. 长沙:国防科技大学出版社,1999.
- [4] 袁胜智,谢晓方,曹建,等. 巡逻攻击导弹作战仿真系统研究[J]. 系统仿真学报,2009,21(14): 4295-4299.
- [5] 王辰辉,杜宏明. 基于 Vega Prime 的液压挖掘机运动仿真[J]. 机械,2009, 36(5): 37-39.
- [6] 黄伟,戴余良,王长湖,等. Creator/Vega Prime 在船舶动力装置视景仿真中的应用[J]. 系统仿真技术,2008, 4(4): 277-281.
- [7] 王海刚,孙俊,郭维勇. 基于 Multigen Creator 和 Vega 的道路交通仿真[J]. 交通与计算机,2007,25(3): 149-151.
- [8] 李建国,周俊武,战凯. 基于 MultiGen Creator/Vega 的矿井视景仿真研究[J]. 有色金属(矿山部分),2009,61(1): 56-58.
- [9] MultiGen Paradigm, Inc. The MultiGen Desktop Tutor [M]. USA: MultiGen Paradigm,2003.
- [10] MultiGen Paradigm, Inc. Creating Models for Simulations [M]. USA: MultiGen Paradigm,2003.
- [11] MultiGen Paradigm, Inc. Lynx Prime Interface (version2.2) [M]. USA: MultiGen Paradigm,2007.
- [12] MultiGen Paradigm, Inc. Vega Prime 2.2 API Tutorial [M]. USA: MultiGen Paradigm,2007.

(上接第196页)

决了 SIP 穿越非对称 NAT 问题。对于 SIP 穿越对称 NAT 问题则需要更进一步的研究,也是下一步的研究工作。

参考文献:

- [1] Schulzrinne R J, Camarillo H, Johnston G, et al. SI-P: Session Initiation Protocol IETF[S]. RFC3261,2002.
- [2] Rosenberg J, Camarillo G, Schooler E, et al. SIP: Session Initiation Protocol[S]. RFC3261,2002.
- [3] 张荣,武波. SIP 协议的应用研究[J]. 计算机技术与发展,2006,16(6): 71-73.
- [4] Rosenberg J, Schulzrinne H, Camarillo G. SIP: Session Initiation Protocol[S]. RFC3261,2002:55-60.
- [5] 张永强,张捍东,赵金宝. SIP 协议栈研究[J]. 计算机

技术与发展,2007,17(11):49-51.

- [6] 刘洋,侯红. 基于 SIP 协议的 IP 电话技术[J]. 计算机技术与发展,2006,16(4):184-186.
- [7] Egevang K, Francis P. The IP Network Address Translator (NAT)[S]. RFC1631,1994.
- [8] Srisuresh P, Egevang K. Traditional IP Network Address Translator(TraditionalNAT)[S]. RFC 3022,2001.
- [9] Srisuresh P, Holdrege M. IP Network Address Translator (NAT) Terminology and Considerations[S]. RFC 2663, 1999.
- [10] 王南,孙保锁,王月平. P2PSIP 系统中 NAT 穿越方案的研究与设计[J]. 计算机技术与发展,2009,19(10):66-69.
- [11] 钟宝荣,杜红,涂继辉. SIP proxy 系统中穿透 NAT 的实现[J]. 计算机技术与发展,2006,16(11):54-55.