• 嵌入式系统工程 •

嵌入式视频监控传输系统的设计与实现

张多英， 申 晨， 刘伟平， 黄红斌
(暨南大学 电子工程系，广东 广州 510632)

摘 要：介绍一个基于嵌入式Linux和达芬奇平台的H.264视频监控传输系统的设计与实现。该系统通过NALU提取模块在达芬奇平台下从其视频压缩后的码流中搜索NAL单元的起始码从而提取出NAL单元，NAL单元经过处理后作为RTP的负载进行传输，而且系统还增加了码率控制模块，根据RTCP的统计信息包SR和RR求得丢包率、时延等情况估计网络拥塞状况来调整编码码率与发送端的发送速率，以获得在视频质量和带宽利用上的平衡。

关键词：嵌入式Linux； 达芬奇； H.264； 实时传送协议； 实时传送控制协议
中图法分类号：TP368.1 **文献标识码**：A **文章编号**：1000-7024 (2010) 04-0724-05

Design and implementation of embedded video monitoring transmission system

ZHANG Duo-ying, SHEN Chen, LIU Wei-ping, HUANG Hong-bin
(Department of Electronics Engineering, Jinan University, Guangzhou 510632, China)

Abstract: The design and implementation of H.264 video monitoring transmission system are introduced. The starting code of NALU in video compressed data at Davinci platform is searched thought the NAL unit distilled module, then pushes video data to RTP thread as load. And in order to get the balance of quality of video and the bandwidth of net, rate control module is added to adjust the rate of video coding and sending based on the information of the SR and RR packet.

Key words: embedded Linux; Davinci; H.264; real-time transport protocol; real-time transport control protocol

0 引 言

视频监控正从传统的安防监控向管理、生产监控发展，并逐步与管理信息系统相结合，达到资源共享，为管理者提供更直观、有效的决策信息。这不仅使人们能以最简便、最逼真、最安全的方式进行远距离实时监控，实现零距离沟通，其在各个领域的广泛应用也为整体社会信息化的发展提供了有力的推动。就其发展进程来说，视频监控系统向着前端一体化、视频数字化、监控网络化、系统集成化方向发展，而数字化是网络化的前提，网络化又是系统集成化的基础。

本系统视频采集用DaVinci平台，连接有摄像头作为采集设备，本地显示设备用于调试，采集后的数据在DaVinci平台上经过视频压缩算法后，以改进后的RTP/RTCP协议进行传输，有效地提高了系统效率。

1 系统设计

1.1 系统架构设计

系统架构主要由3大部份组成：视频采集前端系统、视频服务器、用户IE浏览器，如图1所示。

为了在有限带宽可达到比较清楚的监控效果，本系统采用H.264的视频压缩算法和RTP/RTCP网络传输协议，并以达芬奇开发板作为视频采集前端系统的硬件平台，对采集的图像进行H.264压缩。

1.2 系统工作原理

系统工作原理如图2所示，Davinci(发送端)运行Encode demo程序进行H.264视频图像压缩，并且H.264的NAL层把图像宏块封装后一帧帧地存入缓冲区，通过共享内存的方式把一帧帧的图像数据交由RTP进程，RTP进程在一帧的数据中搜索各个NALU(NAL unit)，并按RFC3984(适合H.264视频的RTP载荷格式)要求，对NAL单元进行分割或聚合，最后以处理后的NAL单元作为RTP负载进行发送。接收端(视频服务器上的RTP接收进程或用户浏览上的播放器插件)根据RTP包头的序列号在接收缓存里对RTP包进行去抖动并把去除RTP包头信息后由H.264进行解码或由服务器把该段视频封装为MOV流媒体格式后保存以便回放。最后接收端周期性地统计出丢包率、数据包到达间隔抖动和回路时间等，以RTCP包反馈给发送端，因此发送端就能根据这些信息判断网络信道状况并控制H.264视频流的发送速率和对H.264编码器的

收稿日期：2009-03-12；**修订日期**：2009-05-30。
基金项目：广东省科技计划基金项目 (2006B11501001)。
作者简介：张多英 (1963－)，男，山东文登人，副教授，研究方向为信号与信息处理； 申晨 (1984－)，男，贵州贵阳人，硕士研究生，研究方向为信号与信息处理； 刘伟平 (1960－)，男，广东韶关人，教授，研究方向为通信与信息系统； 黄红斌 (1976－)，男，广东惠阳人，副教授，研究方向为通信与信息系统。E-mail：sansen777@qq.com

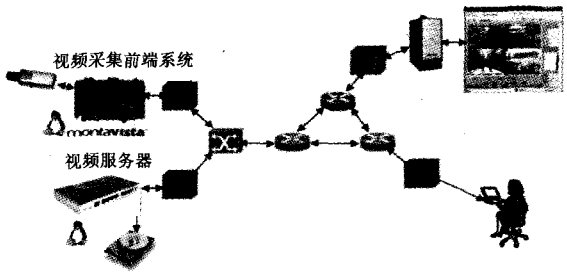

图 1　系统架构

编码控制模块进行码率控制，以减少拥塞的频率，有利于提高网络带宽利用率。

2　H.264 的 NAL 单元提取

H.264 的编码结构在算法概念上分为两层：视频编码层(video coding layer, VAL)和网络适配层(network abstraction layer, NAL)。H. 264 编码器分层结构如图 3 所示，NALU 结构如图 4 所示。

编码后的 H.264 视频序列是由一系列 NAL 单元(NAL unit)所构成。一个 NAL 单元就是一个变长的包括某一类型的语义元素的字节流。比如，NAL 单元可以装载一个图像码片、一个 A/B/C 的数据分割、一个序列或者图像的参数设置等。

由于在达芬奇开发平台下的 H.264 压缩模块(Encode)的输出码流中，每个 NAL 单元前面有 3 个字节的前缀来识别 NAL 单元边，即起始码前缀(0x000001)。所以运行在 ARM 核上的嵌入式 Linux 操作系统的 RTP 进程就可以根据起始码从 H.264 压缩模块的输出码流中提取出各个 H.264 NAL 单元以备发送，即需要通过 Encode 进程中的 video 线程以共享内存的方式获得 H.264 图像帧的数据，并在每帧的数据里搜寻出每个 NAL 单元。提取 NAL 单元流程具体实现如图 5 所示。

3　RTP 传输实现与其负载处理

本系统应用开源的 JRTP 库(C++实现的 RTP 库)来实现基

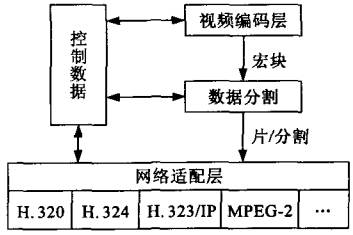

图 3　H.264 编码器分层结构

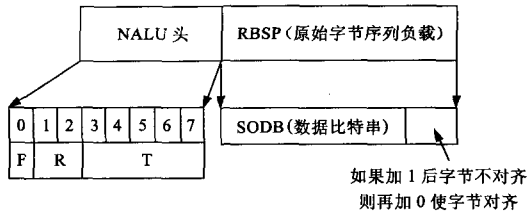

图 4　NALU 结构

本的传输功能，在这基础上根据 RTP 负载的类型进行处理。由于 H. 264 编码器采用 D1 分辨率 (720x576) 进行图像压缩使得编码的后的 NAL 单元较大，因此，这里主要按 RFC3984 标准的 FU-A 的 RTP 载荷格式进行处理，即对 NAL 单元进行分割。FU-A 的 RTP 载荷格式如图 6 所示。

3.1　H.264 视频流的 RTP 打包分析

H. 264 视频流数据在服务器端压缩后，进行 IP 网络传输前，必须打包成适合于网络传输的数据包，采取合适的打包策略对 H.264 视频流在 internet 上的优化传输是十分必要的。通常，适合于网络传输的数据包应满足如下两个条件：

(1)使得载荷数据与头部的开销关系最优化；

(2)使得丢包率尽可能的小。

如果要打包的数据包太大，超过了最大传输单元 MTU(传输层和网络层无需被分割/重组的数据包的最大长度)，那么网络层就要对数据包进行分割，并为每个分割包加上 IP 包头，这样就造成 IP 碎片，接收端在收到这些数据包之后还要对其进行重组，也会加大时延。由于数据包在网络传输过程中丢失的概率总是存在的，一个分割包的丢失会影响到同一个原

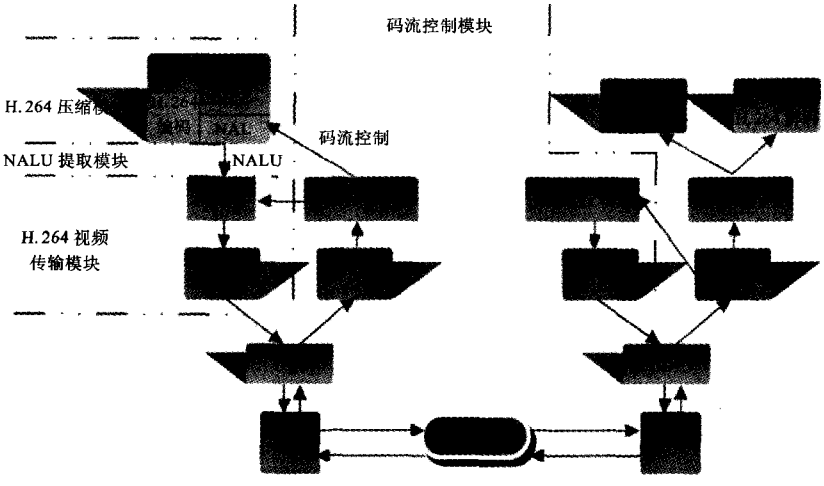

图 2　系统工作原理

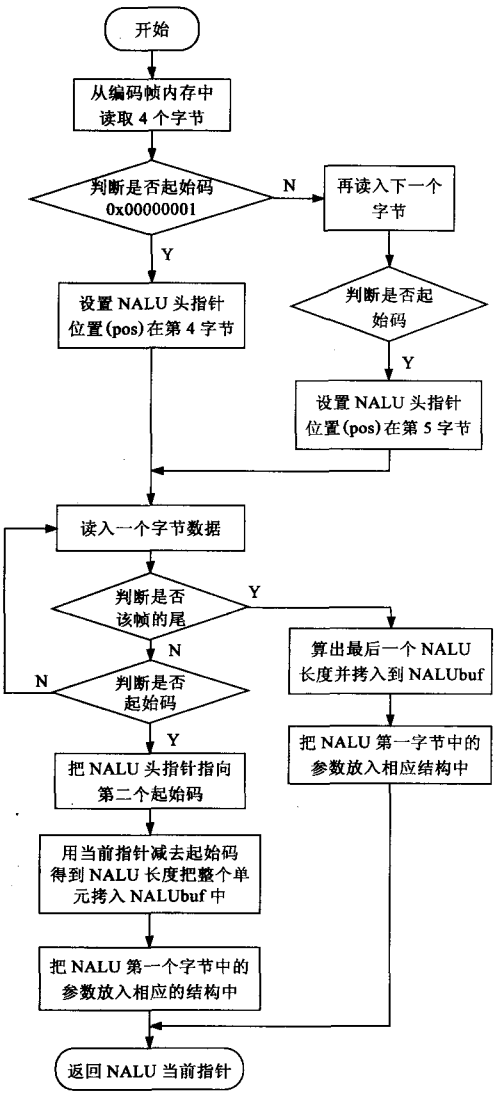

图5 提取NAL单元流程

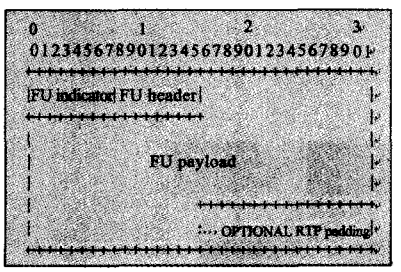

图6 FU-A的RTP载荷格式

始包的其它分割包，所以数据包长度过大，就会降低数据包成功到达的概率，也就降低网络的吞吐量，使网络性能变坏；另一方面，如果要打包的数据包长度过小，将会产生过多的有效信息比较少的小包，这样，会浪费Internet的带宽资源，降低网络利用率。可见，数据包的长度过大或者过小，都会使得网络性能变坏。

不同的视频编码标准有不同的结构和概念，因此对它们进行RTP打包时还要考虑到它们自身的特性，提高封装效率的同时，尽量保证QoS。

3.2 H.264视频流的RTP设计规则

H.264标准提供了网络提取层NAL概念，使得经H.264编码出来的视频流数据能够简单有效地应用到各种各样的网络中。所以在本系统中使用的打包方式就是对单独的NAL单元进行操作，使其满足H.264的RTP载荷规范中明确要求的设计规则：

(1)具有较低的额外开销，因此MTU的大小设计在100-64Kbyte为佳；

(2)易于区分分组的重要性和次要性，而不必对分组内的数据解码；

(3)载荷规范在无需解开比特流的情况下，就能够检测出由于误码所造成的无法解码；

(4)能够支持NAL单元分割到多个RTP数据包中；

(5)能够支持NAL单元重组，在一个RTP包中传送多个NAL单元。

由设计规则可知，NAL单元大小必须满足网络MTU大小，NAL单元首先要封装成RTP数据包，然后封装成UDP数据包，最后封装成IP数据包在网络中进行传输。其中，IP头部占20byte，UDP头部占8byte，RTP头部占16byte，所以，RTP的最大有效载荷长度是1456byte，即每个NAL单元不能超过1456byte。

3.3 NALU分割的实现

在H.264的RTP负载格式中定义了3种不同的负载类型，下面以最为重要的一种类型为例进行实现。为适应以太网MTU的限制，发送端应在发送数据之前先对大的NALU进行分割，以免底层协议(IP)将会对这种大的分组进行拆分导致无法检测数据是否丢失。因此这里采用前文介绍的FU-A方法对NALU进行分割。首先要按RFC3550标准定义FU-A数据结构及实现FU-A的分割NALU操作，其实现代码如下：

```
/*定义分割单元结构*/
typedef struct {
/*FU 指示字节*/
char ind_type:5;   //NALU 类型 == 28，即 FU-A
char ind_nri:2;    //优先级
char ind_f:1;      //禁止位
/*FU 头字节*/
char h_type:5; //NALU 未分割之前的类型
char h_r:1;        //保留位
char h_e:1;        //终止位
char h_s:1;        //起始位
char fu_nalu[IPPACKETSIZE];
}FU_A;
/*被分割 NALU 第一部份*/
void fu_init(FU_A &fu,char *nalu,long nalusize);
/*被分割 NALU 中间部份*/
void fu_mid(FU_A &fu,char *nalu,long nalùsize);
/*被分割 NALU 最后部份*/
void fu_end(FU_A &fu,char *nalu,long nalusize);
```

依据以上 FU-A 的 API 函数对大的 NALU 分割并进行发送，而对于小于阈值的 NALU 则直接传输。

3.4 媒体数据接收

客户端经 UDP 端口接收网络传来的数据包后，首先送入接收缓冲区，然后调用 RTP 接收处理函数进行 RTP 解包处理。RTP 会对数据包的相关信息进行检查，通过检查的包递交给上层进行解码处理，未通过检查的包将直接丢弃。通过合法检查后的报文将报头分离出来，其 RTP 报文有效载荷部分作为数据分组先写入视频数据缓冲区，等待H.264 解码器进行解码。同时根据设置好的反馈时间周期性的发送 RR 报文。

QoS 监控与反馈控制模块的功能就是监测网络拥塞状态，根据 RTP 报头信息分析网络状况，计算出 RTP 报文的丢失率，报文丢失累计、平均延时抖动等相关信息，再生成 RR 报文，反馈给服务器，作为调节编码速率和发送速率的控制信息。

4 码流控制

在多媒体数据实时传输中，H.264 压缩视频流需要在带宽不一致且动态变化的网络上传输，为了充分利用提供的网络资源，并保证用户获得最佳的感觉质量，需要引入码率控制机制。以获得在解码视频质量和带宽利用上的最佳均衡。

码率控制主要是通过引入缓冲区与调节量化参数来实现，而不是对编码帧率不做调整。将编码生成的变码率码流先放到缓冲区，而从缓冲区以一定的速率(根据网络的情况动态调整)将码流输出到信道上，并通过量化参数的调节以保证缓冲区既不上溢也不下溢，即当缓冲比特大于缓冲区的某个阈值时减少量化参数(quantizer parameter，QP)以降低编码器输出比特率；当缓冲比特大于缓冲区的某个阈值时增大量化参数 QP。

码流控制原理如图 7 所示。

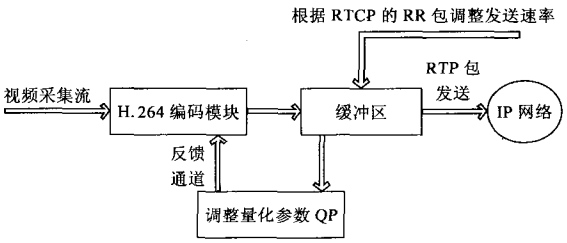

图 7　码流控制原理

4.1 RTP 发送速率控制

由 RTCP 的统计信息包 SR 和 RR 求得丢包率、时延等情况估计网络拥塞状况。设置丢包率门限值 P，当丢包率 p<P 时，网络状态良好，可提高数据发送速率；反之，当网络状态进入拥塞，应降低发送速率。调整发送速率主要是能过调整发送端的发送延时来实现。

4.2 编码器输出比特率控制实现

由于网络情况不断变化，需要对发送端的发送速率不断调整，从而造成缓冲有上溢或下溢的趋势，即当实际缓冲比特占有数超过了缓冲容量 60%，并且实际产生比特数与目标比特数差值越来越大时，则缓冲有上溢的趋势。同样，当实际比特数与目标比特数之差越来越小，并且实际缓冲比特占有数总是低于缓冲容量的 30%的话，缓冲就有下溢的危险。因此，要通过调整编码器的离散余弦变换的量化参数(QP)大小来调整编码器输出比特率。

实际上，量化参数(QP)，即量化步长，它决定了量化器的编码压缩率及图像精度，可以反映了空间细节压缩情况，如QP 小，大部分的细节都会被保留；QP 增大，一些细节丢失，输出比特率降低，但图像失真加强和质量下降。也就是说，QP 和比特率成反比的关系，而且随着视频源复杂度的提高，这种反比关系会更明显。但调整 H.264 编码器的量化参数后不好确定编码器的输出比特率大概是多少，因此，引入了编码码率控制函数来调整编码器的编码质量，该函数也是通过调整H.264 编码器的量化参数来实现，使得编码器最后输出比特率与要求的编码码率基本一致，从而实现码率控制的目的。

对用户来说难以用QP值来衡量编码器的输出比特率，因此达芬奇平台引入了 videoEncodeAlgCreate() VISA API 函数来控制编码器参数，使得用户可以动态调整编码码率来达到调整编码器输出比特率的目的，其原理也是通过调整QP值来实现，只不过是为了使应用更方便才把对 QP 的调整实现给封装起来。

5 实验结果与分析

为了进行流媒体传输测试，需要搭建其测试环境，一般来说，包括：硬件平台环境与软件测试环境；硬件平台环境主要由两台通用主机、达芬奇开发平台(DVEVM)和视频服务器组成。Linux(PC)主机、DVEVM 板、视频服务器通过以太网相连，Windows 主机通过串口跟达芬奇开发平台相连。

首先，在运行 H.264 Encode 程序之前必须通过 loadmodules.sh 脚本加载达芬奇的 DSP/BIOS Link 和 CMEM 内核模块(即双核之间通信与连续物理内存分配模块)。接着，以后台方式运行 RTP 发送程序，并等待视频进行传输。最后，运行 H.264 压缩。然后，在视频服务器及 Linux PC 上运行接收端程序进行接收测试。在用户的 IE 浏览器下的实时监控效果如图 8 所示。

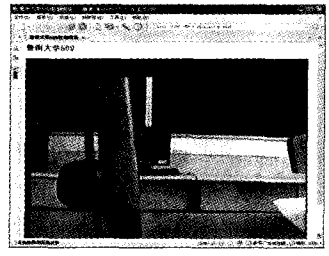

图 8　实时监控测试

由于运动物体最能反应出编码码率与图像失真度之间的关系，下面以一支笔做为测试对象，在 D1(720*576)分辨率情况下，采用不同的编码器输出比特率来测试笔在晃动时的监控效果，图 9 中左图为编码码率为 2Mbit~3Mbit/s 时笔晃动的情况，右边为编码码率是 768Kbit/s 时笔晃动的效果。

通过以上测试，可以知道在D1 分辨率时可对快速运动的

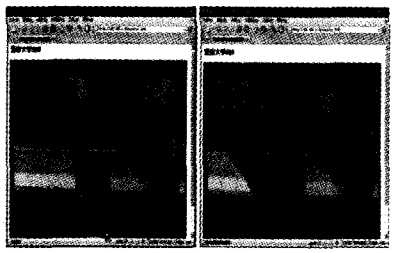

图 9 不同码率下晃动时测试效果对比

物体进行实时在线监控，即使在网络带宽下降到 768Kbit/s 时也可以以 25 帧/s、D1 分辨率对慢速运动进行比较清晰的实时监控。

6 结束语

通过建立测试环境，如开发平台的建立、软件测试环境的搭建。然后进行了实时监控的测试，对各个模块进行了整体测试。最后，在不同编码码率情况下再次进行测试并对其测试效果进行对比，反映出编码码率与图像失真度的关系，从而确定在提供良好的监控效果下用户所需要的网络带宽。

系统具有较高的可扩展性和灵活性，适用于嵌入式Linux环境。测试结果表明，本系统能达到很好地实现监控效果，视频压缩速度快且图像质量好，有很好的实时性，符合嵌入式网络视频监控系统视频数字化、系统网络化、管理智能化的发展趋势。

参考文献:

[1] 刘涛.基于 RTP 的视频流可靠传输[D].郑州大学,2004:13-14.

[2] 杜春晖.监控系统中基于 RTP 的视频处理[J].微计算机信息,2007(23):306-307.

[3] 江东海.基于 ARM 与 Linux 的视频监控系统服务器关键技术研究[D].暨南大学,2006:6-7.

[4] 杨茂林,程伟明.H.264 在分层网络视频监控系统中的应用研究[J].计算机应用研究,2006,23(4):155-157.

[5] Schulzrinne H,Casner S,Frederick R.RTP:A transport protocol for real-time applications[S].RFC 3550,2003.

[6] Wenger S,Hannuksela M M,Stockhammer T.RTP payload format for H.264 Video[S].RFC 3984,2005:8-30.

[7] 陈小平,王皖陵.Linux 下实时流媒体的编程实现[J].安徽工业大学学报,2005,22(7):294-295.

[8] 王原丽,刘建伟.基于 RTP 协议的 MPEG-4 的视频传输系统应用研究[J].计算机与现代化,2007(12):57-58.

(上接第 719 页)

Environment 类指利用环境因素控制角色(role)的应用。环境因素可以从协同工作组与工作流任务两个方面加以考虑。协同工作组层面是利用协同工作组类型限定用户和角色的加入。

例如：编号为 01 的协同工作组，只允许拥有 leader 与 manager 类型角色用户加入。工作流任务是衡量任务需求以及对用户和角色作限定。例如：任务 01 与 03，只允许拥有 super user 与 user 类的用户共同工作，上述两个方面的环境因素，不能同时存在，若同时存在，则会导致角色配置产生冲突。类的属性包括：group 是指设定协同工作组的类型，task 是指设定工作流程中任务条件，factor 是指其它环境因素的定义。

5 结束语

本文提出了一个基于角色扩展的访问控制模型(ARBAC)，利用时间，行为和环境因素对角色加以扩展管理，使得系统管理员能更加弹性化地配置用户角色，以适应协同工作方式的访问控制需求，提高了系统权限访问控制的安全性，降低了角色冲突。同时有助于提高企业的协同工作的用户之间的协同效率，减少企业的成本支出。本研究的后期工作将重点将在委任授权机制方面加以研究，增强 ARBAC 模型的安全性。

参考文献:

[1] 邓集波,洪帆.基于任务访问控制模型[J].软件学报,2003,14(1):76-82.

[2] 覃章荣,王强,欧窝进,等.基于角色的权限管理方法的改进与应用[J].计算机工程与设计,2007,28(6):1287-1284.

[3] 汪厚祥,李卉.D_TRBAC 访问控制模型设计与应用研究[J].计算机工程与设计,2006,27(18):3493-3500.

[4] Anumba C J, Ugwu O O,Newnham L,et al.Collaborative design of structures using intelligent agents[J].Automation in Construction,2002,11:89-123.

[5] Osborn S L,Sandhu R S,Munawer Q.Configuring role-based access control to enforce mandatory and discretionary access control policies [J]. Information System Security, 2003,3 (2): 85-106.

[6] Zhang L,Luo L,Zhang L,et al.Task-role-based access control in application on MIS[C].IEEE Asia-Pacific Conference on Services Computing(APSCC'06),2006.

[7] Kim T,Cera C D,Regli W C,et al.Multi-level modeling and access control for data sharing in collaborative design[J].Advanced Engineering Informatics,2006,20:47-57.

[8] Oh S,Sandhu R.A model for role administration using organization structure[C].Monterey,CA:7th ACM Symposium on Access Control Models and Technologies,2002.

[9] Yu C,Ye D,Wu M,et al.A role-based and agent-oriented model for collaborative virtual environment[C].IEEE International Conference on Systems, Man and Cybernetics, 2005: 1592-1597.