基于 CORBA 的工业机器人软件框架的研究和应用

宋洪军,周风余,李贻斌,贾 磊
SONG Hong-jun,ZHOU Feng-yu,LI Yi-bin,JIA Lei

山东大学 控制学院,济南 250061
Shandong University,Ji'nan 250061,China
E-mail:song_hong_jun@163.com

SONG Hong-jun,ZHOU Feng-yu,LI Yi-bin,et al.Research and application of CORBA-based software framework for industrial robot.Computer Engineering and Applications,2007,43(25):192-196.

Abstract: It is to develop an extensible,scalable and portable software framework for train maintenance using industrial robots. The core is middleware using C++ CORBA,it expatiates at length on the framework's design and implementaion,the testbed includes two Motoman UP6 manipulators and one Pioneer mobile robot,the framework employs two-tier server.System's CORBA IDL are described and explained.Finally,concluding remarks for future works are given.

Key words: middleware;CORBA;industrial robot;train maintenance

摘 要:在国内首次使用中间件技术,面向机车维护机器人生产线,设计开发可扩展,可升级和可移植的软件应用框架。软件框架的核心是 C++ CORBA。试验环境包括 2 台 Motoman UP6 机械手和一台 Pioneer 移动机器人。详细介绍了软件框架的设计和实现,包括采用两层结构实现服务器,框架的技术指标,系统 CORBA IDL 的具体设计和定义。结论部分规划了今后的研究工作。

关键词:中间件;CORBA;工业机器人;机车维护
文章编号:1002-8331(2007)25-0192-05 **文献标识码**:A **中图分类号**:TP242.2

1 引言

火车机车车辆的检测与维护是铁路机务段和机车维护厂的主要任务，检修车间每天需要检测和维护的火车有数十列，每列火车被分解检查,更新部件,然后重新装配后投入运行,工作量巨大。由于车底盘是磨损消耗最集中的部位,因此火车检修主要是针对车厢底盘。机车底盘维护流程大体如下:

(1)在机车承载两侧取出 4 组承载耳并检查承载耳;
(2)在机车承载两侧取出弹簧组并检查弹簧;
(3)翻转机车承载,卸掉交叉杆,检查交叉杆;
(4)卸掉机车承载两端的三角梁;
(5)在四个八字面上去掉垫片,焊接新的垫片;
(6)重新安装机车承载两端的三角梁;
(7)重新安装交叉杆,翻转机车承载;
(8)重新安装弹簧组;
(9)重新安装承载耳。

以上每一步操作持续 5-10 分钟。目前国内所有工作都是人工操作,辅以简单的机械设备,自动化水平非常低。根据上述情况,启动了采用工业机器人流水线实现机车车盘检修自动化的研究。维修线整体设计布局如图 1 所示。

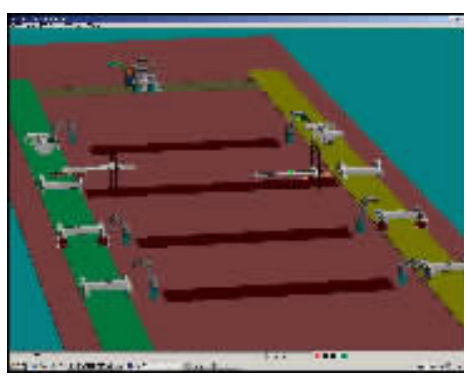

图 1 机器人化的机车车盘检修流程布局

图中流程按逆时针方向进行，右侧工作带是步骤 1-4,左侧工作带是步骤 6-9,中间是步骤 5。步骤 5 配置了 4 台焊接机器人，其他步骤都各配置 1 台机器人，一台移动机器人做为 AGV，负责把弹簧和承载耳从黄色工作带运送到绿色工作带。整个流程包括 13 台机器人,在研发阶段,步骤 3 和步骤 7 使用自行设计制造的特种机器人，使用 Pioneer 先锋做为 AGV,其他步骤使用两台 6 自由度的 Motoman UP6 机械手。

在现场应用中,机器人类型以及工位作业不一样,控制方式和通讯协议也不一样,作业流程是分布异构的。各机器人必

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.60675044);教育部博士点基金(No.20050422035);山东省信息产业发展专项资金(No.2005-56);济南市科技型中小企业技术创新专项资金(No.200505-1051)。

作者简介:宋洪军(1975-),男(汉族),博士研究生,从事工业机器人、虚拟现实和工业中间件的研究;周风余,男(汉族),副教授,主要研究智能机器人;李贻斌,男(汉族),博士生导师,主要研究智能机器人;贾磊,男(汉族),博士生导师,主要研究智能交通和复杂系统控制。

须按一定的节拍工作,以确保整个流程在给定时间内完成。

符合现场应用的软件框架要求能够管理工业机器人,调度指定作业,并为作业的编辑,仿真和实时监控提供有效的视觉手段。根据现场的特性和要求,软件框架采用中间件技术。

目前可用的中间件技术包括 CORBA (Common Object Request Broker Architecture),Microsoft.NET,IBM SOM,SOAP,RTC,Sun's Java/RMI 等。软件框架的系统中间件采用 C++ CORBA,在客户端除了 C++ CORBA,还使用了 Java/RMI。

CORBA 是由 OMG(Object Management Group)提出并维护的独立于供应商的标准协议[1],CORBA 为可移植的分布式计算应用提供了平台无关的编程接口和服务模型。由于独立于编程语言,操作系统平台和网络协议,CORBA 高度适合于分布式应用系统的集成以及在已有系统内开发新应用软件[2]。图 2 展示了 CORBA 机制内的部件模型,这些部件一起实现了 CORBA 的互操作性,可移植性以及其他特性。其中,客户端和驻留在服务器的各种 CORBA 对象通过 ORB(Object Request Broker)互联通讯。ORB 可由不同厂家实现,但是都遵循一致的 CORBA 协议,对于客户端来说都是透明无区别的。

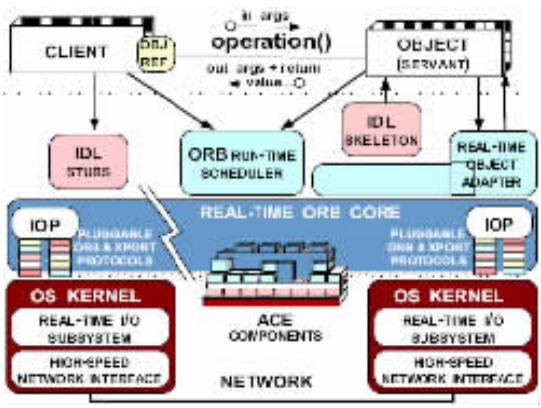

图 2　CORBA 机制内部的部件模型

实时CORBA(RTCORBA:real-time CORBA)协议[3]扩展了 CORBA 核心模型以支持实时架构需要。当前可用的实时 CORBA 是基于 C++和 Java 的实现。实时 CORBA 设计了编程接口,可以在应用程序中配置并控制计算机处理器以及通讯和内存等资源。由于这些特性,软件框架采用了实时 CORBA 提供的若干特性和服务。

首先介绍软件框架的试验环境和系统架构,然后阐述了中间件设计的指标和考虑,展示了系统 IDL 接口的设计和定义,并描述了系统服务器的实现。

2　试验环境

如图 3 所示,软件框架的试验环境是基于局域网的,实际包括 3 个机器人单元,1 台网络计算服务器和若干工控 PC。

机器人单元中有两个是 Motoman UP6 机械手及配套的控制器和本地工控机,该机械手有 6 个自由度,由控制器直接控制,控制器可以通过 RS232 或以太网卡与一台工控 PC 连接,如图 4 所示。

系统服务器的配置如下:

(1)操作系统:RedHat AS3;

(2)CPU:Intel® Xeon™ CPU 3.00GHz;

(3)RAM:2G;

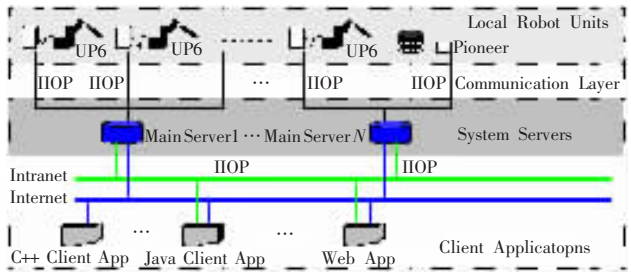

图 3　系统整体架构示意图

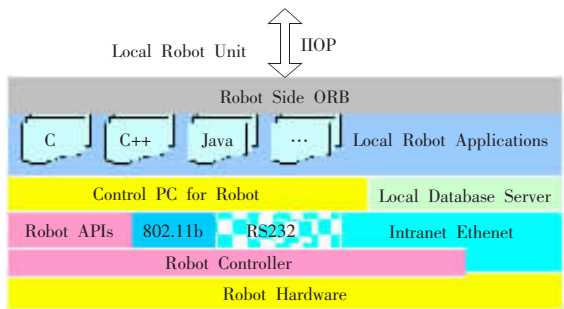

图 4　机器人单元示意图

(4)CORBA:The ACE/TAO 1.4;

(5)数据库:Oracle 9i for Linux。

在客户端研发使用 3 台工控 PC,1 台运行 Linux 操作系统用于 Pioneer 的编程和调试;2 台运行 Windows,其中 1 台运行 Visual C++和 ACE/TAO,用来开发基于 CORBA 的 C/S 方式的客户端应用,另外 1 台安装 JDK 用来开发基于 RMI 的 B/S 方式的客户端应用。图 5 说明了客户端应用结构。

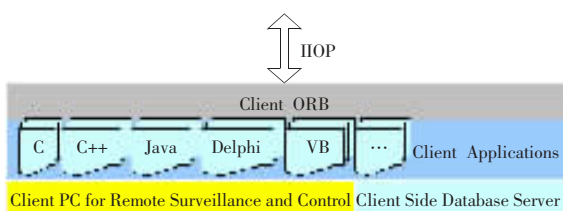

图 5　客户端应用结构

按图 3,图 4,图 5 设计,整个软件框架的通讯都是基于 CORBA IIOP(Internet Inter-ORB Protocol)的。CORBA 协议定义了 GIOP(General Inter-ORB Protocol)做为其互协作的基本框架,但 GIOP 只是抽象概念定义,不能直接应用于 ORB 间的通讯。IIOP 是基于 TCP/IP 的 GIOP 具体实现,软件框架中,客户端应用与系统服务器的通讯,以及机器人工作单元与系统服务器的通讯都是基于 IIOP 的。

3　系统中间件的设计

首先要关注图 3 和图 4,服务器的设计是两层结构,而在[1][4][5][6][7][9]的工作中服务器的设计都是单层结构。

这样设计是因为应用对实时性要求高,如[6]中所论述,在中间件系统中有很多因素导致恶劣的实时性能,包括用户响应,网络传输,CORBA 协议处理,CORBA 服务处理,显示处理等。如果所有 13 部机器人的工作负载以及多个用户的请求处理全部集中于系统主服务器,则系统主服务器就会成为整个系统性能的瓶颈。

为解决此问题,如图 4 所示,由于具体机器人的行为和控制方式都相对固定,针对每部机器人都配置了一台本地工控

PC,设立了 CORBA 环境,并开发了本地服务程序,专门处理机器人的具体操作。这样系统主服务器可以专注于高级的综合任务和事务,如作业调度,远程实时监控,作业仿真,系统日志,客户请求处理等。

系统主服务器通过 RM(Robot Manager)和机器人单元交互,每个机器人在系统主服务器的实时 POA(RTPOA:Real Time Portable Object Adapter)内都驻留着相应的 RM(如图 6 所示)。RM 是主服务器和本地机器人的双重代理,主服务器只和代表机器人的 RM 交互,不去考虑机器人的具体实现;本地机器人也只和自己的 RM 交互,不必考虑其驻留的主服务器的设计和实现。这种关系也体现在图 8 中。

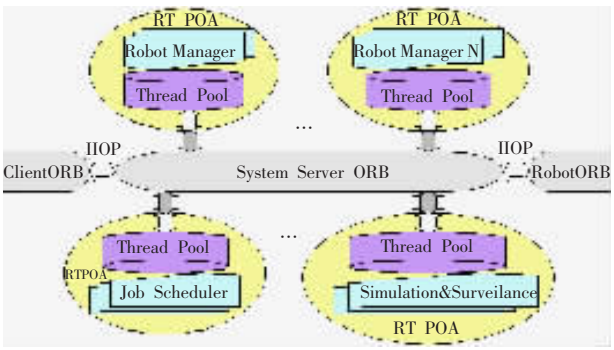

图 6 系统服务器结构

服务器的两层结构设计分离了应用中的低级具体操作和概念级的高层次服务模型,因此提升了系统的升级扩展能力和可移植性。当在系统中增加或移除机器人时,只需要配置对应的 RM 模块而不影响系统整体的概念设计和实现架构,不必考虑机器人具体的操作和相关编码实现。当主服务器移植到新的硬件服务器或运行新的操作系统时,底层的和机器人相关的应用程序都不必在新环境下重新编译或者改动。

在软件框架中采用了 CORBA 标准服务,包括命名服务(Naming Service),实时事件服务(Real-Time Event Service)和实时日志服务(RTEvent based Log Sevice)。

命名服务是基本的 CORBA 标准服务,用以提供透明的中间件服务定位。服务器端驻留并维护着服务对象,并以字符串名字进行广播,客户端可以通过解析这些字符串名字获取所需要的服务对象引用,进而取得所需要的服务,而不必考虑服务对象和服务器的位置和具体实现。

因为要应用于工业现场,采用 CORBA 的实时日志服务为系统提供日志能力。在本框架中,用户的所有操作,请求,相应以及系统内所有的事件,异常,运行状态都可以配置到日志记录中。这些被记录的信息可用来在研发阶段调试系统框架,在现场应用中帮助评估系统性能,协助机器人的示教训练和编码,跟踪用户操作等。

实时事件服务使 CORBA 系统可以交换命名的事件,并没有传统的客户与服务器的界定,而只是发出事件的"供应者"和定制并接受事件的"消费者",事件在供应者和消费者之间的专有事件频道内传输。实时事件服务给应用实现以及数据交换带来了极大的灵活,比如当机器人的空间位姿或作业状态发生变化,就会产生相应的事件,该事件通过事件频道传送到定制事件的消费者,消费者可以是任意的 CORBA 对象。

从远端分布的多个用户来实时的监控多个机器人要求系统较好地利用多线程处理并发。RTCORBA 提供了"线程池"(Thread Pool)机制来支持服务器的多线程编程,实现服务器资源预分配。当服务器启动时,一组线程可以被静态地创建,这些线程随时准备被绑定到客户地请求,可以防止系统运行时线程创建/释放引起的过载,并通过配置服务器的线程数量来确保系统的性能[1]。

为避免低优先级的请求消耗掉所有线程,线程池可以进一步按优先级划分成若干"泳道"(Lanes),这样可以分组管理线程同时保证高优先级的请求不被错过或延误。划分不是绝对的,当高优先级请求所属"泳道"无线程可分配时,可以从低优先级"泳道"借线程使用。

在软件框架中,如图 6 所示,当系统启动时,对应于每一个服务对象(RM,系统日志等)的 RTPOA 就被创建,同时相应的线程池也被创建,划分成若干"泳道"并处于工作待命状态。

针对 RM,线程池按高,中,低三个优先级划分"泳道",一般的作业操作属于低级别请求,状态改变相关的事件属于中级别请求,碰撞和紧急停机属于最高优先级。这样在正常情况下,机器人可执行规定的作业并及时把自身状态事件发布出去,并保证在紧急情况下停止工作和动作。

4 机器人 IDL 文档的设计与定义

软件框架定义了一组机器人对象和接口做为全部工作的基础,所有对象和接口都以 CORBA IDL 定义,确保语言和平台的无关性[8]。这些接口在服务器端具体实现,以伺服程序(Servant)方式体现[4]。

根据 CORBA IDL 定义的对象与接口,图 7 展示了 C++ CORBA 服务器端和不同客户端的实现。在客户端,开发者和用户不必了解对象和接口是如何实现的,只要能获得对象引用,针对可用的接口编程,实现客户端的事务和应用就行了。开发者可以选择 VC++/ACE/TAO,BC++/visiBroker,Java/RMI 等开发套件。在服务器端,软件框架采用 Linux/C++/ACE/TAO 开发。

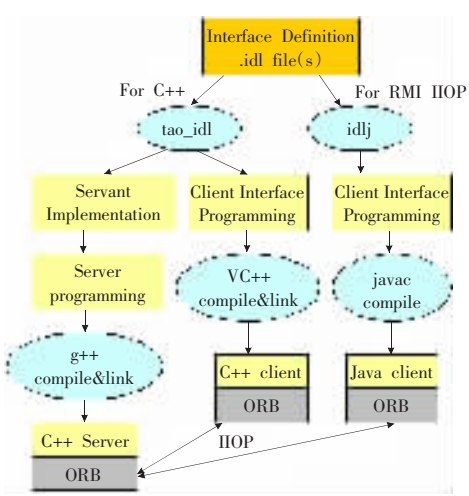

图 7 根据统一的 IDL 定义构建服务器端和客户端

就现场应用而言,机器人应用相关的接口和数据结构可归纳到如下范畴:

(1)异常。异常处理机制是任何软件框架不可或缺的,CORBA 自身定义了 30 个标准系统异常,根据现场应用需要,框架又自定义了若干异常,确保可以详尽获取并管理系统状态。

(2)事件。软件框架采用 CORBA 的实时事件服务做为消息交换的主要方式。在事件定义中主要包含 2 项,唯一标识事

件类型的 ID 和用来承载数据和消息的 CORBA Any 类数据。

(3)作业操作。机车底盘维护流程包括十几项作业,每个作业由 CORBA Sequence 序列来定义。

(4)机器人运动学。每个机器人运行特定的作业,每个作业实际上是运动学动作或运动学矩阵的序列。序列内的每个执行点都是可编辑的,作业序列的本质就是路径规划的过程。因此在本软件框架中,定义了相应的对象和接口,如 UP6 的 DH 表述和正解,逆解。

(5)辅助接口。每个机器人的控制器都有若干辅助函数或接口,如 UP6 的 API 可以协助机器人重置零位或与本地 PC 建立通讯连接等。这些机器人本机 API 都被定义或映射到对应的 IDL 定义中,这样确保在系统中有全局统一的机器人控制接口描述。

系统开发采用不止一种机器人,所以按继承关系来定义机器人对象,如图 8 所示。除了机器人相关对象和接口的定义,还定义了 RM,作业调度,系统日志等服务对象。

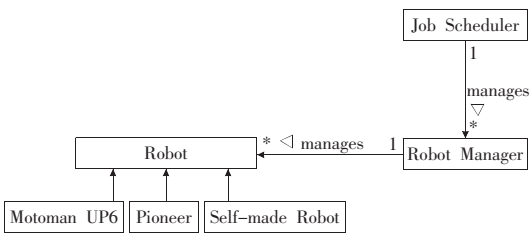

图 8　机器人相关对象的 UML 模型

根据上述设计,下面给出软件框架部分 IDL 定义:

事件结构:

```
struct sysEvent{
    long eventID;
    string eventMsg;
};
```

异常:机器人内部自我碰撞

```
exception InnerCollesion{};
```

异常:机器人与外部发生碰撞,包括机器人之间,机器人与工件,机器人与环境

```
exception OuterCollesion{};
```

机器人类型:机械手,移动机器人

```
enum RobotType{MANIPULATOR,MOBILE};
```

机器人采用的外部通讯协议:RS232 802 无线 Ethernet

```
enum CommProtocol{RS,WIRELESS,ETHERNET};
```

机器人基本概况: 品牌 制造商 供货服务商 机器人型号 机器人类型 机器人编号 通讯协议

```
struct RobotBasic{
    string brand;
    string producer;
    string vender;
    string model;
    RobotType robot_type;
    short robot_num;
    CommProtocol comm_protocol;
};
```

速度类型:线速度,角速度

```
enum SpeedType{LINEAR,ANGULAR};
```

机器人(末端)位姿:

```
struct PositionNormal{
double px; double py;
double pz; double nx;
double ny; double nz;
double ox; double oy;
double oz; double ax;
double ay; double az;
};
```

机器人状态:

```
typedef sequence<PositionNormal> Status;
```

机器人运动控制点:

```
struct ControlPoint{
    SpeedType speed_type;
    double speedrate;
    double duration;
    PositionNormal endPN;
};
```

机器人动作:

```
interface RobotAction{};
```

机器人作业点:

```
struct JobPoint{
  sequence<ControlPoint> ctl_p;
  RobotAction act;
};
```

机器人作业序列:

```
typedef sequence<JobPoint> JobSeq;
```

机器人作业:

```
struct RobotJob{
        JobSeq js;
        string jobname;
        string job_description;
};
```

工业机器人:

```
interface iRobot{
readonly attribute
        RobotBasic basic_description;
readonly attribute
        short degree_freedom;
attribute RobotJob job;
attribute Status status;
void setSpeedTypeOf(in short ctrl_idx,
                    in SpeedType st,
                    in short flag);
void setSpeedRateOf(in short ctrl_idx,
                    in double speedrate,
                    in short flag);
};
```

5　系统服务器的实现

在中间件应用中,服务器端的实现主要是指服务对象的伺服程序(Servant)实现。一个伺服程序就是一组代码集合,可被 POA 用来初始化服务对象, 是服务对象标识的服务的具体实现,可由 C/C++,Java 等多种语言编码。

在机器人单元中, 主要工作是具体实现 Motoman UP6 和 Pioneer 的伺服程序及相关的接口。在系统主服务器中,主要工作是实现 UP6 和Pioneer 对应的 RM 伺服程序, 系统日志伺服

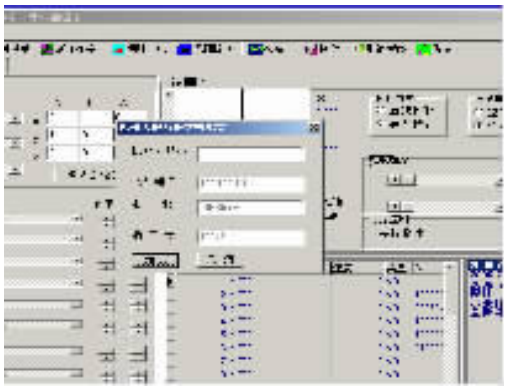

图9 机器人C/S方式客户端界面

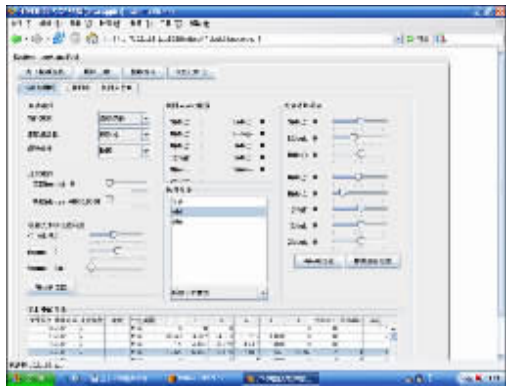

图10 机器人B/S方式浏览器界面

程序,作业调度伺服程序,系统监控伺服程序等。

6 客户端应用

客户端应用开发,结合了传统的C/S方式和最新的B/S方式。客户端程序运行于Windows平台,由C++语言开发。浏览器客户程序采用Java RMI实现,通过tomcat来发布服务。

图9是C/S方式客户端C++程序界面,用户首先通过CORBA命名服务来选择指定的机器人,然后针对选择的机器人完成各种操作,如编辑和仿真作业或修改机器人运行参数。

图10是B/S方式浏览器界面,功能和操作与C/S方式基本相同。

7 结语

阐述了基于中间件的机器人控制软件框架的设计和实现,虽然软件框架是为火车机车维护研发的,但也可以用于其他复杂分布式异构环境中的机器人应用。今后的工作将专注于多机器人复杂作业调度和移动机器人的传感信息处理。

(收稿日期:2007年4月)

参考文献:

[1] Bottazzi S,Caselli S,Reggiani M,et al.A software framework based on real-time CORBA for telerobotics system[J].Digital Object Identifier,2002,3(30):3011-3017.

[2] Henning M,Vinoski S.Advanced CORBA® Programming with C++[M].[S.l.]:Addison Wesley,1999.

[3] OMG.Real-Time CORBA specification[M].Object Management Group,Inc,2002:20-78.

[4] Kuo Yuan-hsin,MacDonald B A.A distributed real-time software framework for robotic applications[C]//Proceedings of the 2005 IEEE International Conference on Robotics and Automation Barcelona,Spain,April 2005:150-155.

[5] Pan Liandong,Huang Xinhan.Implementation of a PC-based robot controller with open architecture[C]//Proceedings of the 2004 IEEE International Conference on Robotics and Biomimetics August 22-26,2004,Shenyang,China,2004:211-216.

[6] Fernández-Madrigal J-A,Cruz-Martín E,Cruz-Martín A,et al.Adaptable Web interfaces for networked robots[J].Digital Object Identifier,2005:3441-3446.

[7] Chen Qijun,Geng Haixia,Woo Peng-Yung.Research on and pure Java realization of a web-based mobile robot system[C]//Proceedings of The American Conlrol Conference,Denver,Colorado June 4-6. 2003:276-278.

[8] Utz H,Sablatn¨og S,Enderle S,et al.Miro-middleware for mobile robot applications[J].IEEE Transactions on Robotics and Automation,2002,18(4):159-165.

[9] Marin R,Sanz P J,Sanchez J S.A very high level interface to teleoperate a robot via Web including augmented reality[J].Digital Object Identifier,2002,3:2725-2730.

(上接161页)

通信的应用(如IDS的远程管理),解决了单向通信的传输可靠性问题,具有较好的应用价值。

由于BAATM跳过了TCP/IP协议栈传输层,直接在应用层进行ARQ,因此其通用性依赖于应用层是否按BAATM机制来传输数据,可能对用户的透明性不够好。若能将TCP/IP协议栈传输层的传输机制按BAATM改进,将解决BAATM对用户的透明性问题。这是下一步的研究内容。

(收稿日期:2007年1月)

参考文献:

[1] Comer D E.Internetworking with TCP/IP Volume 1:principles,protocols,and architectures [M].4th ed.Beijing:POST & TELECOM Press,2002.

[2] Postel J B.RFC 793 transmission control protocol[S],1981.

[3] Postel J B.RFC 768 user datagram protocol[S],1980.

[4] Sollins K R.RFC 1350 the TFTP Protocol(Revision 2)[S],1992.

[5] Mogul J C,Deering S E.RFC 1191 path MTU Discovery[S],1990.

[6] RFC1722 RIP version 2 protocol applicability statement[S],1994.

[7] Braden R T.Extending TCP for transactions-functional specification [S].Internet Draft,1992.

[8] Braden R T.RFC 1122 requirements for Internet hosts-communication layers[S],1989.

[9] Braden R T.RFC 1123 requirements for Internet hosts-application and support[S],1989.

[10] Clark D D.RFC 813 window and acknowledgement strategy in TCP[S],1982.

[11] Clark D D.The design philosophy of the DARPA Internet protocols.Computer Communication Review,1988.