文章编号：1005-8451（2009）02-0051-04

自律分散协议及其在Linux上的实现

谭珍珠
（西南交通大学 电气工程学院，成都 610031）

摘 要：自律分散协议是实现自律分散系统的基础和关键。简单介绍了自律分散系统的概念和自律分散协议的主要内容；介绍了利用开源的Linux操作系统及其TCP/IP协议栈来扩展自律分散协议的方法。

关键词：自律分散系统；ADP；TCP/IP；实现
中图分类号：TP39 文献标识码：A

Autonomous decentralized protocol and it' s implementation on Linux

TAN Zhen-zhu
(School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: Autonomous Decentralized Protocol was the foundation and key of Autonomous Decentralized System. It was simply introduced the concept of Autonomous Decentralized System and autonomous decentralized protocol, and then given a method to implement the protocol with Open Source Linux System and TCP/IP protocol stack.

Key words: ADS; ADP;TCP/IP; implementation

自律分散系统（Autonomous Decentralized System, ADS）是建立在广播、多播通信基础上、每个节点自律运行的一种系统。自律分散协议（Autonomous Decentralized Protocol，ADP）是系统中各个节点通信的基本协议，是构建自律分散系统的基础。自律分散协议可以在标准的TCP/IP协议上实现。在协议层次上，可以看做TCP/IP参考模型的应用层协议。故自律分散协议可以在任何具有完整TCP/IP协议栈的系统上实现。

Linux操作系统具有完整的TCP/IP协议栈。TCP/IP协议族在Linux网络体系结构的网络协议层。Linux的网络实现是以BSD为模型，TCP/IP

收稿日期：2008-09-16
作者简介：谭珍珠，在读硕士研究生。

图2 AutoCAD ActiveX程序主界面

5 结束语

详述论述了AutoCAD ActiveX后，对利用C#对AutoCAD进行二次开发在轨道交通平面信号布置图绘制的应用进行了分析和实现。这样绘制的平面信号布置图在联锁表的自动生成的程序开发中还有进一步的优势，因为它的主要部分都是以图块和图块属性的方式给出所以在图形信息提取的时候更加容易和方便。

参考文献：

[1] 李长勋. AutoCAD ActiveX二次开发技术[M]. 北京：国防工业出版社，2005.
[2] 王永辉，胡青泥，李红彩. AutoCAD二次开发方法的研究[J]. 计算机系统应用，2007（3）：94-96.
[3] 梅 松. 基于ActiveX技术的联锁表自动生成软件[J]. 铁道通信信号，2007（6）：9-10.
[4] 鹿士杰，袁泽虎，王 娟. 面向对象的技术在CAD二次开发中应用[J]. 湖北工学院学报，2002，17（2）：38-39.

协议栈是基于AF_INET套接字地址族的。因为自律分散协议是TCP/IP上的应用层协议，所以大部分协议的实现都可以通过AF_INET所支持的套接字服务类型来实现。

1 自律分散系统与自律分散协议

1.1 自律分散系统

1.1.1 概念

自律分散系统是由许多自律的子系统（节点）组成的一个复杂的系统。其中某些子系统可能处于故障状态、正在进行改进或维修。子系统具有平等性和局部性是自律分散系统的基本特点。

1.1.2 结构

自律分散系统的基本结构如图1。

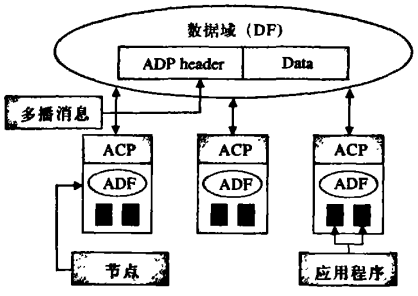

图1 ADS结构示意图

（1）节点：可以看成ADS的子系统，是构成ADS的最基本自律单元。在物理网络中,它对应于计算机、智能设备或其他硬件。每个节点都包含有自己的ACP (Autonomous Control Processor)和应用程序模块,能独立地从数据域中提取信息并进行内部处理，同时又主动地以广播方式向数据域发送处理结果及其他内部信息。

（2）数据域（DF）：是ADS中信息传播的空间。从物理概念上讲,它相当于网络或存储器。一方面所有节点主动地向数据域发送信息,同时各个节点又根据自己所需从数据域中取走信息以完成内部模块的功能。

（3）节点数据域（ADF）：可以看成是DF延伸到节点内的部分，其中流动的是节点内部应用程序（UP）所需的数据。

（4）自律控制处理器（ACP）：它的大部分功能是基于ADP协议来实现的，ACP在节点接收消息时起到一个过滤器的作用，只有注册了相应交易码的数据才能进入节点。

（5）交易码（TCD）：在数据域中流动的信息是基于ADP协议的多播消息，在其ADP header都包含一个交易码TCD（Transaction CoDe）来标志其属性,各个节点就是通过识别交易码来决定自己是否需要此信息的。这种基于交易码的通讯方式保证了每个子系统的自律信息发送和自律信息接收。即每个子系统不必知道信息来源和目的地之间的关系，因而它实现了每个子系统的局部性。当一个程序所需的数据到达数据域时，由系统自动启动该程序。这种方式称为数据驱动机制。

数据域和广播的通讯方式是实现ADS系统模型的关键概念。数据域、交易码通信、数据驱动是自律分散系统的3大特征。

1.2 自律分散协议

自律分散协议（Autonomous Decentralized Protocol）是实现自律分散系统的网络通信协议。每个自律分散子系统或节点都要运行ADP。ADP是TCP/IP网络协议中的应用层协议。如图2。

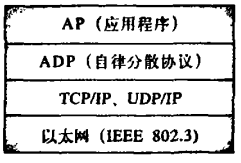

图2 ADP协议在TCP协议中位置

ADP协议功能的主要内容有多播（Multicast）通信、生存信号、故障信号、测试支持和点对点通信等。其中多播通信和传输生存信号是ADP协议的基本功能。本文主要介绍这两个基本协议功能的实现。

（1）多播通信协议

多播通信是一种一对多通信。是自律分散系统的最基础的通信方式。多播通信的地址是多播组。多播组是一个数据域内建立的一组节点，一个数据域内可有多个数据域，每个节点必须属于至少一个多播组。多播组号是标识一个数据域内的多播组的唯一标识。

多播组内的节点只接受附有自己要求交易码（TCD）的消息。因为这种方法不指定任何目标地

址，所以通信可以扩充。

对多播组的管理有分配多播组号和UDP端口数、加入和离开多播组以及创建多播发送端口。

播组号从1~255，多播组0被分配给存活信号。每个多播组号对应一个UDP端口号；节点可以加入和离开一个多播组；发送多播消息的源端口号可以为每个多播组分配不同的值，也可以一个数据域内分配一个源端口值。

（2）生存信号

节点向数据内其他节点间周期性地发送的一种消息。当节点接收到此消息，则表明发送节点是“存活”的。当在指定的发送周期之间没有接收到节点的生存信号，则认为节点“死亡”。也就是表明节点处于死机等其他严重异常状态。

（3）ADP的协议数据单元结构

ADP的协议数据单元结构如图3，有ADP header和Data两部分构成。ADP header长度为64bytes，其内容包括消息源地址、目标地址、tcd、消息优先级等。

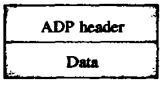

图3　ADP的消息结构

2　ADP的Linux实现

ADP是位于TCP/IP上的一层应用层协议。ADP的各种协议内容可以封装成一些ADP应用接口，以供应用程序调用。为了在Linux上支持多播，在编译配置内核时，需要激活“IP：multicasting”选项。

2.1　多播通信

ADP的多播通信是基于TCP/IP的IP多播通信。多播组号实际上就是一个IP多播地址（D类地址），对多播组的操作实际上就是对IP多播地址的操作。存活信号的IP多播地址固定为224.0.0.1，此地址连接局域网内全体具有多播功能的主机；多播组对应的端口号：55001~55255（这是在线模式的端口号）。与多播组相关的数据还有套接字，当加入一个多播组、接收多播组消息和离开多播组时都要利用套接字。故本地节点的ACP需要维护一个多播组号MGN的链表。为了方便对多播组的操作，链表中一个节点的结构体struct MGN，声明如下：

```
typedef struct MGN {
char mc_addr[14];
unsigned char mgn;
int mc_port;
int socket;
}
```

其中，mc_addr为MGN对应的点分十进制多播IP地址；port为对应的端口号；socket为与此MGN关联的套接字描述符；mgn为多播组号的值。

2.1.1　加入一个多播组

封装的接口函数为：int ads_add_mgn(struct mgn)；函数参数为要加入的多播组号。

函数的实现：根据参数多播组号mgn找到IP多播地址；然后创建一个数据包类型（datagram）的socket，并把此socket值付给MGN->socket；把IP多播地址和多播组对应的端口号填入多播组地址信息里；绑定到此socket；调用函数setsockopt()设置套接字选项IP_ADD_MEMBERSHIP，至此节点已加入多播组mgn。

2.1.2　离开一个多播组

接口函数声明：int ads_leave_mgn（struct MGN mgn）；

函数实现比较简单：先调用函数setsockopt()设置套接字选项IP_DROP_ MEMBERSHIP，则套接字脱离多播组；然后关闭套接字close(soc-ket)，把MGN链表中相应的多播组号节点删除；

2.1.3　接收多播消息

节点加入多播组以后就可以接收多播消息了。

接口函数：int ads_recv（struct MGN mgn, const void *adf_buffer,size_t lenth）；

函数参数adf_buffer指向用来保存接收到的消息的地址，其为所有应用程序（AP）所共有的固定值，可以看做节点内部的数据域；Lenth为消息长度（字节）。Lenth的类型size_t为无符长整形。

实现：函数主体调用Linux的UDP消息接受函数recvfrom()或者是recvmsg()接收多播消息；然后再提取出数据包的TCD，如果在TCD注册表

里能找到此TCD就把此消息放到adf_buffer指定的缓存里（节点数据域）。

一个节点可能有不止一个应用程序（AP），节点接收到的多播消息可能有很多个AP需要（比如存活信号），那么在节点数据域里的消息就必须在每个需要此消息的AP提取（复制）之后才可以删除。为了实现此功能，我们在TCD注册表里添加一项：每个TCD码后面加上注册的次数，即AP每注册一次就把此数加1。AP提取相应的次数后就把此消息删除。注册表是一个链表，其节点的结构示意如下：

```
typedef struct Tcd {
TcdType code;
int regNum;
struct Tcd *next;
}
```

2.1.4 发送多播消息

发送多播消息需要指定消息内容、发送地址和TCD等。多播发送是基于UDP数据报的，由于UDP数据报的最大长度受到设备的MTU（最大传输单元）限制，这里我们假定MTU=1500 bytes，则相应的多播消息的最大长度=1408 bytes。当消息的长度大于此时就要分片发送。

函数声明：ads_send(struct MGN,void *buf, int len,int tcd);

其中参数MGN为多播消息的目的多播组；buf、len分别为要发送的多播数据的存放地址和长度；tcd为交易码；

发送多播消息的过程，首先创建一个UDP类型的socket，填入多播地址和端口号；

然后把检查多播数据的长度是否超过最大限制，如果超过则分片发送，并在ADP header中设置响应的数据片序列号。

最后调用socket函数sendto()把消息（片）发送出去。关闭套接字。

2.1.5 注册TCD码

当AP需要带有特定TCD的信息时，就必须向节点数据域注册TCD。

函数：int ads_regTCD(int TCD);

注册方法很简单，就是向TCD注册表的链表里追加加一个TCD节点。

以上函数的返回值在函数调用正常的情况下为0，异常情况下根据错误情况分别定义相应的错误号。

2.2 AP使用多播通信的过程

一个节点进行多播通信的大致流程如图4。

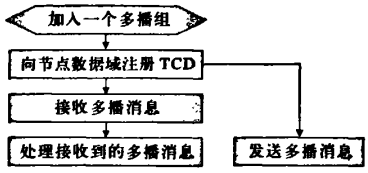

图4 节点多播通信的流程

其中存活信号的发送是在后台一直运行的。

3 结束语

自律分散系统是最近几年发展起来的一种新的系统结构，在电力自动化系统、铁路运输调度自动化系统等领域具有广阔的应用前景，国内关于ADS的研究主要集中在理论上。本文介绍了在Linux上实行ADP基本协议的一种方法。自律分散协议是自律分散系统各个节点之间的通信协议，是构建自律分散系统的基础。ADP协议是TCP/IP协议层之上的一种应用层协议，由于Linux具有完整的TCP网络协议，并且开源。在Linux上实现ADP协议是实现自律分散子系统非常可靠有效的方法。对进一步研究自律分散系统具有重要的意义。

参考文献：

[1] Distributed Manufaeturing Architecture Committee, Japan FA Open Systems Promotion Group. Manufacturing Science &Technology Center MSTC/JOPl\01.SpecifieationsforAutonomousDeeentralizedProtoeolR3.0!S].Tokyo:httP://www.mste.or.jp/bs/report/R30adp.PDF.1999.

[2] 赵顺玲，于胜龙，吴德昌，谭永东. 传统监控系统的不足及ADS解决方案[J]. 电力自动化设备，2003（3）：2.

[3] Kinji Mori .Trend of Autonomous Decentralized Systems[EB/OL]. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04), 2004.

[4] 倪 瑾，宋中山. 基于Linux网络协议栈实现及应用[J]. 中南民族大学学报（自然科学版）. 2005，(24)：4.