DCOM 协议在网络冗余环境下的应用

郑　耿,刘国平
(中国科学院自动化研究所 复杂系统与智能科学实验室,北京 100080)
(zgrong @sina. com)

摘　要:文中从网络冗余环境下 DCOM 组件程序冗余功能设计中的一个基本问题出发,深入研究了 DCOM 协议,分析出问题发生的原因并通过网络数据包捕获解码分析的方法进行了证实,最后提出了该问题的一种解决方法"组件标识拆分"。

关键词:DCOM;组件;网络冗余

Analysis and Application of the DCOM Protocol in a Network Redundancy Environment

ZHENG Geng, LIU Guo-ping
(*Laboratory of Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, Beijing* 100080, *China*)

Abstract: In this paper, a basic design problem of DCOM component program in a network redundancy environment is presented. The DCOM protocol is studied and the reason of the problem is analyzed, which is also proved by analyzing the network data packets. Finally, a solution for the problem named as "component identity split", is proposed.

Key words: DCOM; component; network redundancy

1　前言

目前,组件化已经成为程序设计的主流。它的设计思路就是参照硬件的开发方式,将复杂的应用程序分解成各种小的功能单一的组件模块,然后再将它们组装成所需的应用程序。这种开发方式能够显著提高软件质量和软件开发效率。为了实现这样的应用软件,组件之间必须要遵守共同的规范。为此,Microsoft 公司提出了 COM/ DCOM 技术规范。

组件对象模型(COM)定义了组件(即 COM 对象)开发中的各种定义以及组件之间交互的标准,但只适合于单机情况。为适应分布式计算的需要,Microsoft 公司在 COM 的基础上提出了分布式组件对象模型(DCOM),可以支持网络中不同计算机上的组件之间的相互通信。由于 DCOM 是对 COM 进行的无缝扩展,并刻意屏蔽了底层网络细节。对于普通的分布式应用来说,DCOM 程序的开发同 COM 程序开发没有多大差异,只需要了解网络环境和单机环境的某些不同,如网络延时,安全等,以及知道如何进行程序配置即可。目前关于 DCOM 方面的文章,如文献[1][2]等,大多是介绍基于 DCOM 的分布式应用程序的设计和实现,没有也无需深入研究 DCOM 协议。本文对 DCOM 协议进行了深入研究,所提出的问题和解决方法,对于网络冗余环境下的 DCOM 程序设计具有较好的借鉴作用。本文所描述的网络冗余环境,指的是网络冗余但没有主机冗余的场景。这在目前的工程应用中还是非常普遍的。

2　问题的提出

在一些关键性的分布式应用中,因为网络故障而造成应用中断是不能容忍的。一般对于这样的应用,都需要提供两个以上彼此独立的网络进行冗余,并且要求应用程序也要提供相应冗余功能,即在一个网络发生故障的情况下能尽快通过备用网络进行通信,维持程序正常运行。

在采用 DCOM 技术开发提供冗余功能的应用程序时,就自然想到能否依靠系统内在的 DCOM 服务来实现网络故障诊断,通信切换以及恢复等功能而无需在上层应用程序中考虑。具体思路就是应用程序通过两个网络与同一个组件进行通信,程序拥有同一个组件在不同网络的接口指针。在一个网络的接口指针方法调用失败后,程序可以通过另一个网络的接口指针来调用同样方法来维持程序正常运行。但在现实测试中却发现这种思路是无法工作的。

一个典型的测试情况如下:两台机器均运行 Windows 2000 中文版,每台机器都配备两块网卡连接在彼此独立的网络中(简称为 A 网和 B 网),在一台机器上运行 DCOM 组件程序,而另一台运行客户程序。在程序正常运行的情况下,人工断开某个网络来模拟网络故障并考察程序的运行情况。测试用的 DCOM 组件是个简单的标准组件。客户端程序的主要思路是程序定时性的通过 A 网创建组件实例并运行该组件方法,然后释放,接下去再通过 B 网创建组件实例并运行该组件方法,然后释放。这样就可以考察一个网络出现故障后程序能否通过使用另一个网络得到的接口指针来正常运行。示

范代码如下:

```
//A 网创建组件
COSERVERINFO ServerInfoA = {0, L"192.168.100.45", 0, 0};
hr = CoCreateInstanceEx(CLSID-Server, NULL, CLSCTX-ALL,
    &ServerInfoA, 1, &multiQIA);
//使用方法,然后释放
pServerA = reinterpret-cast < IServer * > (multiQIA.pItf);
pServeAr -> GetStatus(&nVal);
pServerA -> Release();
//B 网创建组件,网络地址不同
COSERVERINFO ServerInfoB = { 0, L"10.1.1.1", 0, 0 };
hr = CoCreateInstanceEx(CLSID-Server, NULL, CLSCTX-ALL,
    &ServerInfoB, 1, &multiQIB);
//使用方法,然后释放,基本同 A 网
```

通过人工模拟网络故障可以发现奇怪现象,即一个网络断开有可能会造成通过另一个网络创建组件得到的接口方法调用同样失败,而且这种现象是有点随机性的。如 A 网被断开,B 网正常工作,则 A 网的 CoCreateInstanceEx 将会失败,B 网的 CoCreateInstanceEx 是成功的,但返回的接口指针进行的方法调用将会有可能失败,失败的错误都是 RPC-S-SERVER-UNAVAILABLE。但此时 B 网通信是正常的。这种奇怪现象就需要进一步对 DCOM 协议进行研究。

3 DCOM 协议原理及问题原因分析

DCOM 协议是一种规定面向对象的远程过程调用的应用层协议,也被称为 ORPC。它是建立在 DCE RPC 规范之上的扩展,指明了如何对一个远程对象进行调用以及对象引用如何表示、通信和维护。DCOM 协议基本上使用标准的 RPC 数据包,但附加上了一些专用信息,具体数据包定义等可参见 DCOM 协议说明[3]。

下面详细分析一下客户远程创建 DCOM 组件以及使用组件方法时的流程。客户调用 CoCreateInstanceEx 来远程创建组件,DCOM 实现需要通过 RPC 调用服务器端的 IRemoteActivation::RemoteActivation 来要求激活指定的组件。函数返回指定组件的 OXID(对象输出标识符)和所需接口的 IPIDs(接口指针标识符)。每次 RemoteActivation 调用激活同样的组件时,服务器端可以自由选择是返回一个新的组件还是返回一个老的组件。Windows 的实现则是返回一个老的组件。

客户在使用接口方法时,首先需要根据已有组件的 OXID 来获得调用所需的 RPC 通道,然后再通过该通道进行接口方法的 RPC 调用。每台机器的 SCM(COM 服务控制管理器,程序为 RPCSS.EXE 或 RPCSS.DLL)都实现了一个 OXID 解析器,保存和提供本地客户要通信的远程对象所必需的 RPC 绑定字符串。RPC 绑定字符串确定了通信协议、地址、端口、选项等网络通信信息。OXID 解析器实现的 RPC 接口为 IOXIDResolver。它维持一张缓存表,可以映射 OXID 到 RPC 字符串。客户需要解析 OXID 时,OXID 解析器首先是在本地缓存表里查找对应的 RPC 字符串,如没有找到才同远程服务器的 OXID 解析器联系来获取对应的 RPC 字符串。在服务器端可以为一个 OXID 分配多个 RPC 通道,即可以映射到多个 RPC 字符串,对应于不同的网络协议。

OXID 具体解析过程如下[4]:

(1) 如果客户端的 COM 运行库第一次使用该 OXID,则将会请求本机的 OXID 解析器来解析该 OXID。

(2) 如果客户端的 OXID 解析器没有该 OXID 的 RPC 绑定字符串缓存,则进行 RPC 调用 IOXIDResolver::ResolveOxid 来请求服务器的 OXID 解析器来返回该 OXID 的 RPC 绑定字符串。否则直接返回本地缓存的绑定字符串。

(3) 服务器的 OXID 解析器在本地表里查找并返回客户端 OXID 解析器所需要的 RPC 绑定字符串。如果没有找到,但是服务器也支持客户指定的通信协议,则从该协议实现中分配资源并更新本地表,增加该 OXID 对应的 RPC 绑定字符串,并将它返回给客户端。

(4) 客户端的 OXID 解析器更新本地缓存表,只保存最好的 RPC 绑定字符串,以便以后使用,再将它返回给客户进程的 COM 运行库。

(5) 客户使用得到的绑定字符串同组件进行绑定,然后才可以进行该组件的方法调用。

可以看出,由于 Windows 的 DCOM 实现对于多个客户通过 CoCreateInstanceEx 来创建同样标识的组件时只会创建一个实例,因此给客户返回的都是同样的 OXID。由于 OXID 对应的 RPC 绑定字符串是被客户端缓存的,因此存在缓存失效的可能。对于本文所讨论的网络冗余环境来说,客户端为 OXID 保存的 RPC 绑定字符串只能是一个网络的,要不是 A 网,就是 B 网。假如保存的是 A 网的,那么即使是通过 B 网创建的组件的方法调用,可实际上还是通过 A 网来通信调用的。这就解释了一个网络断开有可能会造成通过另一个网络创建组件得到的接口方法调用同样失败的现象。这种现象的随机性同 OXID 解析以及缓存更新的时间相关。

4 网络数据包分析

本节通过捕捉和分析网络数据包的方式来进一步证实上节的分析。NAI 公司的 SnifferPro 程序是一个功能强大的网络协议分析软件,能分析绝大多数网络协议。在两台机器上都运行 SnifferPro,设置一台机器上的 SnifferPro 对一个网络进行监听,另一台就对另一个网络进行监听。数据包过滤的主要设置如下:监听地址填入实际对应网卡的硬件地址,监听的协议只选 TCP。这样可以大大减少要分析的数据。然后运行上述的测试程序,再对捕获的数据进行分析。

目前 SnifferPro 还无法直接针对 DCOM 协议进行分析,只能分析到 RPC 这一层,具体 DCOM 数据包解码只能人工进行,具体解码分析考虑到篇幅在此省略。对于 DCOM 数据包,SnifferPro 都显示为 MS/DCE RPC 数据包。下面只对相关信息进行说明。测试程序中的每次 CoCreateInstanceEx,组件方法调用和接口释放都会进行一次 DCOM 调用。以图 2 为例,从具体解码中可以发现,B 网中的每次 DCOM 调用的数据包(如序号 11 和 12 为一次完整的 DCOM 调用)实际上是 B 网的 CoCreateInstanceEx 调用,但没有捕获到接下去的组件方法调

用和接口释放的数据包,实际上它们都是在 A 网上发送的。如图 1 所示,序号 29 至 38 为测试程序中一次示范代码执行发送的数据包,经过具体解码可以发现,序号 29 和 30 为 A 网的 CoCreateInstanceEx 调用,31 和 32 为 A 网组件方法调用,33 和 34 为 A 网组件接口释放,35 和 36 为 B 网组件方法调用,37 和 38 为 B 网组件接口释放。也就是说,此时客户端的 OXID 解析器保存的 RPC 绑定字符串指定的网络地址是 A 网地址(192.168.100.45),因此对于 B 网创建的组件的使用照样是通过 A 网来通信的,进一步证实了第三节进行的分析。同样可以对两网的 CoCreateInstanceEx 调用返回的数据包进行分析(如图 1 的 30 和图 2 的 12),可以看出两网 CoCreateInstanceEx 调用所创建的组件的 OXID 是相同的。

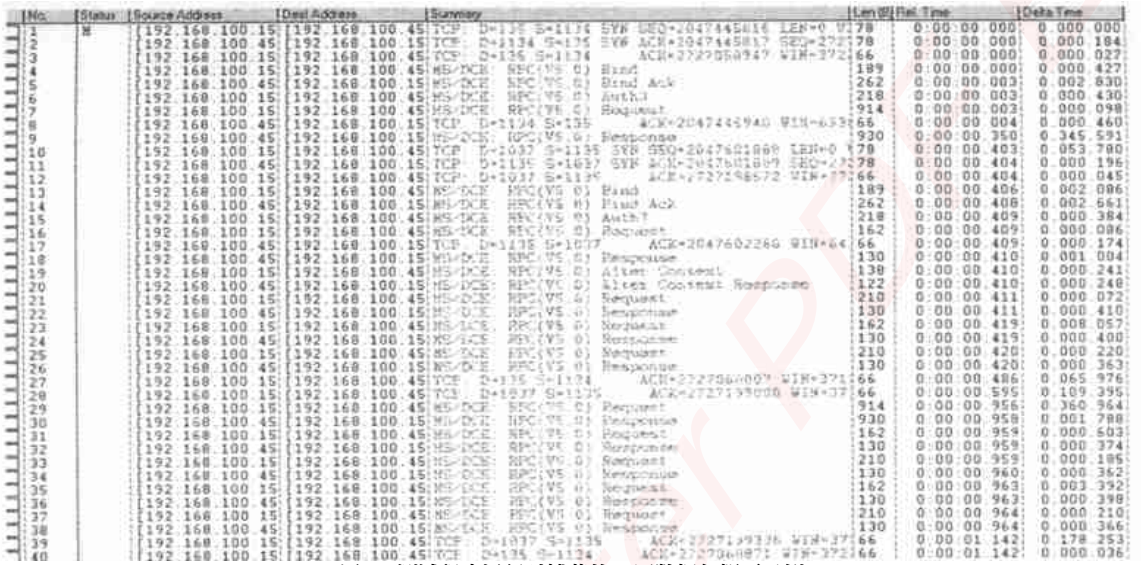

图 1　测试程序运行时捕获的 A 网数据包概要示例

图 2　测试程序运行时捕获的 B 网数据包概要示例

5　问题的解决方法

从上面分析可以得出普通的 DCOM 组件使用方式是无法做到直接支持网络冗余的。一种巧妙的解决方法就是组件标识拆分,为每个使用的组件对象建立一个镜像对象,即两者只有名字(CLSID 和 ProgID)不同,而其他功能方面是完全相同的。应用程序可以将这两个组件当成同样的组件来使用,而 DCOM 服务却将它们看成是两个完全不同的组件。如果应用程序在不同网络使用镜像组件,DCOM 服务会为每个组件保存对应于不同网络的 RPC 绑定字符串,不会出现两个组件使用同一个 RPC 绑定字符串的情况,从而解决上面提出的问题。程序的具体思路就是先通过两个网络来创建两个组件(一对镜像组件),程序拥有不同网络的镜像组件接口指针。在一个网络的接口指针方法调用失败后,程序可以通过另一个网络的接口指针来调用同样方法来维持程序正常运行,实现了要求的应用冗余功能。整个程序的设计和编码都和普通 DCOM 应用的设计和编码也没有多少区别。具体例子限于篇幅原因在此省略。

6　结论

本文首先提出了网络冗余环境下 DCOM 组件程序设计中的一个基本问题,即一个网络断开有可能会造成通过另一个网络创建组件得到的接口方法调用失败。接着对 DCOM 协议进行了深入研究,分析出问题发生的原因是在于 DCOM 服务的 OXID 解析机制。然后通过网络数据包捕获解码分析的方法对分析出的原因进行了证实,最后提出了该问题的一个巧妙解决方法,组件标识拆分,对于类似环境下的 DCOM 程序设计具有较好的借鉴作用。

参考文献

[1]　袁春阳,李琳,等. 基于 DCOM 的分布式入侵检测系统模型的设计和实现[J]. 计算机工程,2003,29(6):139～141.

[2]　熊伟,孙娜,等. 基于 DCOM 的分布式可重组系统的研究与实现[J]. 计算机应用,2002,22(11):62～65.

[3]　Microsoft Corporation. Distributed Component Object Model Protocol - DCOM/1.0[EB]. MSDN, 1997.

[4]　Eddon G, Eddon H. Understanding the DCOM Wire Protocol by Analyzing Network Data Packets[J]. Microsoft System Journal, 1998,(3).

[5]　潘爱民. COM 原理与应用[M]. 北京:清华大学出版社,1999.

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现
44. 基于 PCI 总线的 MPEG2 码流播放卡驱动程序开发
45. 基于磁盘异或引擎的 RAID5 小写性能优化
46. 基于 IEEE1588 的时钟同步技术研究
47. 基于 Davinci 平台的 SD 卡读写优化
48. 基于 PCI 总线的图像处理及传输系统的设计
49. 串口和以太网通信技术在油液在线监测系统中的应用
50. USB30 数据传输协议分析及实现
51. IEEE 1588 协议在工业以太网中的实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法

20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现
25. WindML 中 Mesa 的应用
26. VxWorks 下图形用户界面开发中双缓冲技术应用
27. VxWorks 上的一种 GUI 系统的设计与实现
28. VxWorks 环境下 socket 的实现
29. VxWorks 的 WindML 图形界面程序的框架分析
30. VxWorks 实时操作系统及其在 PC104 下以太网编程的应用
31. 实时操作系统任务调度策略的研究与设计
32. 军事指挥系统中 VxWorks 下汉字显示技术

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现

24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现
33. LINUX ARM 下的 USB 驱动开发
34. Linux 下基于 I2C 协议的 RTC 驱动开发
35. 嵌入式下 Linux 系统设备驱动程序的开发
36. 基于嵌入式 Linux 的 SD 卡驱动程序的设计与实现
37. Linux 系统中进程调度策略
38. 嵌入式 Linux 实时性方法
39. 基于实时 Linux 计算机联锁系统实时性分析与改进

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现
19. 基于 WinCE 的嵌入式图像采集系统设计
20. 基于 ARM 与 WinCE 的掌纹鉴别系统

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植
15. 基于 PowerPC 嵌入式内核的多串口通信扩展设计
16. 基于 PowerPC 的多网口系统抗干扰设计
17. 基于 MPC860T 与 VxWorks 的图形界面设计
18. 基于 MPC8260 处理器的 PPMC 系统
19. 基于 PowerPC 的控制器研究与设计

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问

11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计
22. ARM CortexM3 处理器故障的分析与处理
23. ARM 微处理器启动和调试浅析
24. 基于 ARM 系统下映像文件的执行与中断运行机制的实现
25. 中断调用方式的 ARM 二次开发接口设计
26. ARM11 嵌入式系统 Linux 下 LCD 的驱动设计
27. Uboot 在 S3C2440 上的移植

Hardware：

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究
16. 基于 UEFI 的 Application 和 Driver 的分析与开发
17. 基于 UEFI 的可信 BIOS 研究与实现
18. 基于 UEFI 的国产计算机平台 BIOS 研究
19. 基于 UEFI 的安全模块设计分析

20. 基于 FPGA NiosⅡ的等精度频率计设计
21. 基于 FPGA 的 SOPC 设计
22. 基于 SOPC 基本信号产生器的设计与实现
23. 基于龙芯平台的 PMON 研究与开发
24. 基于 X86 平台的嵌入式 BIOS 可配置设计

Programming:

1. 计算机软件基础数据结构 - 算法
2. 高级数据结构对算法的优化
3. 零基础学算法
4. Linux 环境下基于 TCP 的 Socket 编程浅析
5. Linux 环境下基于 UDP 的 socket 编程浅析
6. 基于 Socket 的网络编程技术及其实现