一种基于 IPv6 的流媒体传送方案研究与实现

贾崇敏,沈苏彬
(南京邮电大学 计算机学院,江苏 南京 210003)

摘　要:近年来,基于 RTSP 协议的流媒体应用逐渐得到普及,并且网络和网络应用正处在从 IPv4 向 IPv6 过渡时期,而 RTSP 1.0 并未规定如何在 IPv6 网络中传送流媒体。为了解决流媒体在 IPv6 网络环境下的传送问题,文中提出一种基于 IPv6 的流媒体传送方案,在研究 IPv6 对流媒体传送增强特性的基础上,通过分析 IPv4 与 IPv6 套接口的差异,运用 IPv4/IPv6 兼容套接口编程技术,重新封装开源流媒体项目 live555 底层数据通信模块 GroupSock,并修改上层传送协议,最终实现流媒体在 IPv6 网络下的传送。测试结果表明,该方案能够支持 IPv6 网络环境下的流媒体传送。

关键词:IPv6;流媒体;Socket;live555
中图分类号:TP393　　**文献标识码**:A　　**文章编号**:1673-629X(2013)11-0194-05
doi:10.3969/j.issn.1673-629X.2013.11.048

Research and Implementation of a Streaming Transport Scheme Based on IPv6

JIA Chong-min, SHEN Su-bin
(College of Computer, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: Recently, RTSP-based streaming media applications gained popularity gradually, network and network applications transiting from IPv4 to IPv6, however, RTSP 1.0 does not specify the way how to transmit streaming medias in IPv6 networks. An IPv6-based streaming transport scheme was presented for solving the problem of transporting streams in IPv6 networks. On the basis of researching the enhancements aimed at streaming transport by IPv6, through analyzing the difference between IPv4 and IPv6 Socket, it was implemented by resealing the open source project live555 underlying data communication module GroupSock and modifying the upper transport protocol using IPv4/IPv6 compatible Socket programming technology. The test results showed that the scheme can support streaming transmitted in IPv6 networks.

Key words: IPv6; streaming media; Socket; live555

0　引　言

随着计算机网络和多媒体技术的快速发展,流媒体应用发展迅速。目前流媒体相关技术成为研究热点,但仍然存在着许多问题。其中,亟待解决的问题之一是:由于当前互联网是基于 IPv4 协议,随着 Internet 用户和应用的不断增加,IPv4 已暴露出地址空间严重不足、数据传输缺乏质量保证、数据安全性难以保证和对组播功能支持有限等问题,这在一定程度上限制了流媒体应用的进一步发展。

能否在 Internet 上高质量地传输流媒体数据取决于三个方面的因素[1]:网络带宽的不断提高;数据压缩技术的优化;适用于实时数据传输的新技术和新协议。IETF 于 20 世纪 90 年代中期设计了 IPv6[2],它除了提供更大的地址空间之外,还可以提高网络的整体吞吐量、改善服务质量(QoS)、提供更好的安全性保证、支持即插即用和移动性、具有更好的组播支持,另外,IPv6 加入了流的概念,通过采用流标签的方法提供一种更为高效的处理数据分组的机制。因此,基于 IPv6 流媒体传送技术具备重要的研究价值和良好的应用前景,是未来流媒体发展的方向。

文中首先分析了 IPv6 针对流媒体传送的增强特性,包括 IPv6 流特性与 IPv6 组播技术;其次,研究流

收稿日期:2013-01-20　　**修回日期**:2013-04-23　　**网络出版时间**:2013-08-28
基金项目:江苏省科技支撑计划项目(BE2009157)
作者简介:贾崇敏(1988-),男,硕士研究生,研究方向为计算机网络;沈苏彬,研究员,博士生导师,CCF 会员,研究方向为计算网络、下一代电信网以及网络安全。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20130828.0837.026.html

媒体数据在IPv6网络下的传送技术,包括流媒体传送协议和IPv4 Socket接口到IPv6 Socket接口的转换方法;最后,基于开源流媒体库live555,实现了IPv6流媒体传送方案。

1 IPv6对流媒体增强特性

IPv6的出现解决了传统IP的瓶颈问题[3],为了提升互联网网络性能,IPv6定义了"流"的概念以提供更好的服务质量(QoS),并增强了组播,为更高质量的流媒体实时传送服务提供了底层协议支持。

1.1 IPv6流特性

IPv6为流媒体传送提供了更好的服务质量(QoS)。IPv6定义了"流"(Flow)的概念,所谓流是在互联网上从特定源节点发往特定目的节点(单播或组播)的一系列数据分组,而源节点要求数据分组在传送路径上的中间路由器保证所指定的QoS。表1所示,RFC2460[2]将IPv6报头加入了8 bits的"流量类型"和20 bits的"流标签"字段,"流量类型"将IP分组的优先级划分为256个等级,对于那些需要特殊QoS的流类型,可以在IP数据包中设置相应的优先级,路由器根据IP包的优先级分别对这些数据进行处理。

表1 IPv6报头结构

版本	流量类型	流标签
有效载荷长度	下一个头部	跳数限制
源地址		
目的地址		

"流标签"的使用目的是使源点标识一个流,网络在源和目的端之间建立一条有特殊属性需求的伪连接。例如,源主机的一个进程到目的主机的一个进程的分组流有严格的延迟要求与带宽要求。这时可以预先创建"流"并分配流标签。当该流标签字段的分组出现时,路由器在其内部表中找出它所需的特殊需求并做相应处理。

1.2 IPv6组播

IPv6协议增强了对组播技术的支持,避免了在IPv4网络中部分路由器不支持组播的情况。随着因特网的用户数目的急剧增加,以及多媒体通信业务需求不断提高,有更多的业务需要组播来支持。IPv6组播提供了一种由源点通过一次发送操作把同样的分组副本发送到多个终点的方法,即一对多的通信。与单播相比,在一对多通信中,组播在整个网络的任何一条链路上只传输单一的数据包,它提高了数据传输的效率,降低了主干网拥塞的可能性,这对流媒体信息的传送来说意义尤其重大。IPv6组播组的目的主机可以在一个局域网内或者同一私有网络的不同节点上甚至分布在整个互联网上,这为网络流媒体服务提供了更大的灵活性。总之,IPv6组播技术进行流媒体数据传输,能有效利用网络带宽利用率,大大减少网络和服务器负载。

如表2所示,IPv6重新定义了组播地址,前缀为FF,标志位字段用于确定是永久分配地址"0"还是临时分配地址"1",范围字段用于确定组播传送的距离。IPv6协议使用组播接收者发现协议MLD,替换了IPv4中的IGMP的功能。另外,IPv6网络对组播路由协议和组播树的生成都有良好的支持。总之,IPv6增强了组播技术,它为流媒体数据在互联网中的一对多传输提供一种高效、灵活的方式。

表2 IPv6组播地址格式

8 bits	4 bits	4 bits	112 bits
FF	标志	范围	组ID

2 方案分析

2.1 流媒体传送技术

流媒体技术是指连续的音频和视频数据在服务器端经过压缩编码后,用户可以通过网络一边下载一边观看视频节目,而无需下载整个文件的实时播放技术、方法和协议的总称。当前主要的流媒体产品有三个:Windows Media、RealSystem和QuickTime。此外,还存在许多优秀的开源流媒体解决方案,如live555[4]、Darwin[5]等。

流媒体传送和控制协议主要包括RTP、RTCP、RTSP和SDP。RTP[6]是一种提供端对端传输服务的实时传输协议,用来支持在单播和组播网络服务中传送实时数据;它由紧密相关的两部分组成:负责网络上流媒体数据实时传输的RTP和负责反馈控制、监测管理传输质量的实时传输控制协议(RTCP)。RTSP[7]是一种客户端与服务器之间多媒体传输描述协议,它被用于控制媒体流的传输,为多媒体服务扮演"网络远程控制"的角色。会话描述协议(SDP)主要是为会话通知、会话初始以及其他形式的多媒体会话初始化等操作提供多媒体会话服务。

2.2 IPv4向IPv6 Socket编程接口转换方法

IPv4与IPv6协议套接字编程接口差异主要体现在地址结构、地址转换函数、域名转换函数、加入和退出组播组、特殊地址和一些特定参数等方面[8-10]。

第一,IPv4地址使用struct in_addr和struct sockaddr_in定义,IPv6的地址使用struct in6_addr和struct sockaddr_in6定义。为了实现对IPv4和IPv6地址结构的兼容,将struct sockaddr_in修改为通用地址结构struct sockaddr,并根据sa_family字段来确定实际的地

址类型和长度。

第二,IPv4 通过下列函数完成字符串到网络字节序形式的相互转换:inet_aton(strptr, addrptr)和 inet_ntoa(inaddr)。IPv6 则通过下列函数完成字符串到网络字节序形式的相互转换:inet_pton(af, src, dst)和 inet_ntop(af, src, dst, cnt)。由于 inet_pton 和 net_ntop 是地址兼容的,根据参数 af 完成 IPv4 或 IPv6 地址的转换。

第三,IPv4 通过下列函数完成主机名到地址的解析:gethostbyname(hostname)。其中入口参数 hostname 为主机名,函数返回的结果存放在 struct hostent 结构中。IPv6 中引入了 getaddrinfo(),既可用于 IPv4 也可用于 IPv6,返回一个 addrinfo 的结构指针。

第四,不管是 IPv4 还是 IPv6,组播组的加入和退出都是使用下列函数[11]:setsockopt(sockfd, level, optname, optval, optlen)。其中,sockfd 为 socket 的值,optlen 为 optval 对应数据的长度,level 为 IPPROTO_IPV6 或 IPPROTO_IP。当 optname 设为 IP_ADD_MEMBERSHIP 时表示加入组播组,设为 IP_DROP_ MEMBERSHIP 时表示退出组播组。

第五,IPv4 的环回地址为 127.0.0.1,定义为 INADDR_LOOPBACK;对应的 IPv6 的环回地址::1,定义为 in6_addr_loopback;IPv4 通配地址定义为 INADDR_ANY,而 IPv6 通配地址定义为 in6addr_any。

2.3 基于 IPv6 流媒体传送方案分析

由于 RTSP 1.0 协议在 RFC2326 中并未定义对 IPv6 协议的支持,而 RTSP 2.0 目前还处在草案状态,并未最终通过评审成为标准。因此,为了在当前 RTSP 1.0 协议环境下实现对 IPv6 协议的支持,就需要对 RTSP 1.0 协议进行修改。

为了实现基于 IPv6 协议的数据传送,首先需要底层协议的支持,Linux 内核中已经加入了 IPv6 协议栈。其次,需要 Socket 应用编程接口 API 的类型转换,即将基于 IPv4 的 Socket 编程接口转换为基于 IPv6 的 Socket 编程接口。最后,需要修改应用层相关参数,与 IP 地址相关类型转换为 IPv6 类型。

live555 的 GroupSock 模块用于实现数据包发送和接收,它被设计用以支持组播,但也支持单播通信。主要实现了对底层 Socket 编程接口的封装,通过分析 GroupSock,根据协议无关性原理[12]和 IPv4 向 IPv6 Socket 编程接口转换方法,重新修改对底层 Socket 编程接口的封装,实现上层 RTSP 和 RTP/RTCP 协议对 IPv6 Socket 编程接口的支持。live555 的 LiveMedia 模块主要实现了 RTSP、RTP/RTCP,修改 LiveMedia 模块中对底层 Socket 封装函数的调用,包括参数类型、地址结构等,实现上层协议对 IPv6 地址类型的支持。

3 系统实现

该方案基于开源流媒体库 live555,目标是在 live555 流媒体库上实现基于 IPv6 的流媒体传送功能。主要针对底层通信库 GroupSock,实现 IPv6 Socket 编程接口的封装、内部模块的数据结构调整等,主要有以下两种实现方案:

方案 1:重新实现底层 socket 调用的封装,对于涉及到数据传送的类,重新定义新的类。当使用到这些与协议相关的类时,用参数 family 来区分协议版本。这种方法思路清晰,但需要对原有的类库进行扩充,改动较大,不易实现。

方案 2:重新实现底层 socket 调用的封装,对于涉及到数据传送的类,不定义新的类,而是修改类的定义或实现,用 family 参数控制协议的版本。这种方法改动量较小,仅仅对原有库中类的结构进行改动而不扩充新的类的定义。

为了尽量复用当前模块,同时保证对外接口变化较小,现采用方案 2。关于 IPv6 的实现,live555 依赖于操作系统对其提供 Socket 调用的支持,向上层应用程序提供调用接口。文中基于 Linux Socket 编程接口进行实现,为了实现 IPv6 流媒体传送功能,必须对网络层相关协议进行修改,重点需要修改 GroupSock 等底层 Socket 封装类,上层协议需要修改 IP 地址结构类型,下面给出对 live555 的主要改进。

3.1 GroupSock 模块的改进

(1)修改 IP 地址数据结构,添加对 IPv6 地址结构的类型定义;定义地址类型兼容结构体,将 IP 地址存储结构设计为联合体类型,可以在减少存储空间的前提下兼容支持两种地址类型;参数 family 用于标识 IP 类型。代码实现如下:

```
typedef struct in_addrnetAddressBits6
typedef struct in6_addr netAddressBits6;
typedef struct{//地址兼容类型结构定义
    union{
        netAddressBit saddr;//IPv4 地址
        netAddressBits6 addr6;//IPv6 地址
    } addrtype;int family;//地址类型 AF_XXX
}netAddr;
```

(2)对 IP 地址转换函数的封装接口进行改进,加入对 IPv6 地址转换的支持。实现方法:变量 family 判断地址类型,AF_INET 表示 IPv4 地址类型,AF_INET6 表示 IPv6 地址类型;根据地址类型分别进行处理,将字符串形式的 IP 地址转化为网络字节序地址并存储在 netAddr 类型的变量中。代码描述如下:

```
our_inet_addr(const char * cp,netAddr * ap){
    if(AF_INET== ap-> family)//IPv4
        inet_pton(AF_INET,cp,&s6_addr);
```

```
    else if(AF_INET6 == ap-> family)//IPv6
        pton(AF_INET6,cp, &s_addr));
}
```

(3)添加对 IPv6 套接口地址初始化宏定义 MAKE_SOCKADDR_IN6,实现对 IPv6 套接口地址结构的初始化工作,代码描述如下:

```
#define MAKE_SOCKADDR_IN6(var,adr,prt) \
    netAddressBits6var;\
    var.sin6_family = AF_INET6;\
    var.sin6_addr = (adr);\
    var.sin6_port = (prt);\
```

(4)类 GroupSock 是对网络接口的封装,用于收发数据包,其中定义了对特定 socket 进行数据读写的方法。将 GroupSock 类中与 IP 类型相关的参数修改为地址兼容类型 netAddr。并在相应的函数实现中根据 family 的值分别处理。函数 changeDestinationParameters 用于修改的目的地址,将参数中的目的地址类型修改为 netAddr;添加对 IPv6 地址比较算法的支持。

(5)对 socket 连接方式进行修改。函数 socketJoinGroup 用于加入指定的组播组,socketLeaveGroup 用于离开指定的组播组。修改点是根据输入参数组播地址 groupAddress 的 family 的值,IPv6 地址将内部结构体 ip_mreq 修改为 ipv6_mreq 数据结构,同时应用 IPv6 组播操作模式,将函数 setsockopt 的原有的输入参数 IPPROTO_IP 和 IP_ADD_MEMBERSHIP 修改为 IPPROTO_IPV6 和 IPV6_ADD_MEMBERSHIP,实现对 IPv6 组播组的支持。针对 socket 创建函数 setupDatagramSocket 和 setupStreamSocket,添加 family 参数,根据 family 创建 IPv4 或 IPv6 套接字。

(6)添加 IPv6 特殊地址的定义。IPv6 全 1 地址的定义:#define IN6ADDR_NONE_INIT {{{0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff}}}。定义 IPv6 类型的接收接口地址和发送接口地址,netAddressBits6 ReceivingInterfaceAddr = IN6ADDR_ANY_INIT; netAddressBits6 SendingInterfaceAddr = IN6ADDR_ANY_INIT。

3.2 上层协议库 liveMedia 模块的改进

(1)RTSPClient 定义了流媒体客户端类,成员变量 fServerAddress 表示服务器地址,将其原有类型 netAddressBits 修改为通用地址类型 netAddr,并修改赋值方式。函数 describeURL 用于描述 URL 的 SDP,修改点是在生成 SDP 描述信息时,将 IP 地址修改为 netAddr 类型,并将 SDP 的长度增加至 128 位。

(2)RTSPServer 定义了流媒体服务器类,成员变量 fClientAddr 表示客户端地址,将原有类型 struct sockaddr_in 修改为套接字通用地址类型 struct sockaddr,用于支持 IPv6 地址。重新实现 RTSPClientSession 成员变量 fClientAddr 的初始化。将函数中所有与 IP 地址相关的类型都修改为地址兼容类型 netAddr,并在函数实现中分别进行处理。

(3)RTCPInstance 类是对 RTCP 通信的封装。RTCP 主要功能是统计包的收发,为流量控制提供依据。成员函数 incomingReportHandler1、unsetSpecificRRHandler 和 setSpecificRRHandler 是 RTCP 数据处理句柄函数,改进部分是将其地址类型修改为 IPv6 兼容类型 netAddr,并将函数 fSpecificRRHandlerTable 的 Add、Remove 和 Lookup 操作修改为支持 IPv6 的方式。

(4)SDP 格式的改进。ServerMediaSession 类用来处理会话中的描述,它包含多个音频或视频子会话描述,函数 sdpLines 用于生成 SDP 描述信息,根据 URL 中的 IP 地址类型生成相应的 SDP,若为 IPv4 时 SDP 相应字段格式为"c=IN IP4 IPv4Address",IPv6 时相应字段的格式为"c=IN IP6 IPv6Address"。

4 测试与分析

4.1 测试环境

(1)采用 Ubuntu 10.04,需加载 IPv6 模组:modprobe ipv6,配置 IPv6 全球单播地址,其中服务器端设置为:2001:470:23:13::103,客户端设置为:2001:470:23:13::102。

(2)重新编译 live555,生成流媒体测试服务器 live555MediaServer。流媒体播放器基于 live555 库进行数据传送,Xvid 视频解码,Qt 实现图像显示。媒体源以文件形式存放在流媒体服务器端,采用 MPEG-4 编码格式。

(3)利用网络封包分析软件 Wireshark 对网络报文进行截获并分析。

4.2 可行性测试及结果分析

首先,开启流媒体服务器,等待客户端的请求;开启 Wireshark;运行流媒体播放器,输入 URL:rtsp://2001:470:23:13::103/test.m4e。捕获报文结果如图 1 所示,流媒体播放器运行效果如图 2 所示。

No.	Source	Destination	Protocol
30	2001:470:23:13::102	2001:470:23:13::103	TCP
31	2001:470:23:13::103	2001:470:23:13::102	TCP
32	2001:470:23:13::102	2001:470:23:13::103	TCP
33	2001:470:23:13::102	2001:470:23:13::103	RTSP
36	2001:470:23:13::103	2001:470:23:13::102	RTSP/SDP
38	2001:470:23:13::102	2001:470:23:13::103	RTSP
40	2001:470:23:13::103	2001:470:23:13::102	RTSP
42	2001:470:23:13::102	2001:470:23:13::103	RTSP
43	2001:470:23:13::103	2001:470:23:13::102	RTSP
45	2001:470:23:13::103	2001:470:23:13::102	RTP
49	2001:470:23:13::103	2001:470:23:13::102	RTP
50	2001:470:23:13::103	2001:470:23:13::102	RTP
51	2001:470:23:13::103	2001:470:23:13::102	RTP

图 1 Wireshark 捕获报文图

图 1 所示的报文序列描述了建立 RTSP 会话的过

程:前3个分别是在建立TCP连接三次握手过程的报文;报文33、38和42分别是客户端向服务器发送的DESCRIBE、SETUP和PLAY请求报文;而报文36、40和43分别为服务器向客户端发送的应答报文;之后服务器开始发送RTP包;其中报文36为RTSP/SDP,服务器向客户端发送的SDP描述信息。

图2 流媒体播放器运行效果

测试结果显示,数据在传送过程中所使用的网络层协议已经转化IPv6,即支持IPv6网络环境下的流媒体传送功能。由图2结果可知,采用IPv6协议的流媒体系统运行稳定、图像流畅,可以满足一定的用户需求。

5 结束语

文中对IPv6流媒体特性进行阐述,分析IPv4向IPv6套接口编程过渡技术和基于IPv6的流媒体传送方案,基于开源live555项目进行实现,给出具体的实现方案,并对该方案进行可行性测试。测试结果表明,该方案完全支持IPv6网络环境下的流媒体传送功能,可以满足用户对图像的性能需求。

参考文献:

[1] 梅 艳.基于IPv6的流媒体传输技术及其在校园网中的应用[J].通信学报,2006,27(11A):31-34.

[2] Deering S, Hinden R. Internet protocol, version 6 (IPv6) specification[S]. IETF RFC 2460,1998.

[3] 徐宇杰.IPv6深入分析[M].北京:清华大学出版社,2009:4-16.

[4] Live555streaming media[EB/OL]. 2011. http://www.live555.com.

[5] Darwinstreaming server[EB/OL]. 2008. http://dss.macosforge.org.

[6] Schulzrinne H, Casner S, Fredrick R, et al. RTP: a transport protocol for real-time applications[S]. IETF RFC 3550, 2003.

[7] Schulzrinne H, Rao A, Lanphier R. Real time streaming protocol (RTSP)[S]. IETF RFC 2326,1998.

[8] 汪 浩,严 伟,任茂盛.基于规则的IPv4源程序到IPv6源程序的移植方法[J].计算机工程,2004,30(23):25-27.

[9] Wang Yonghua, Xing Yuelin. Transition of Socket applications from IPv4 to IPv6[C]//Proc of international conference on computer engineering and technology (ICCET). [s. l.]:[s. n.],2010:75-78.

[10] Cheng Yuanhang, Han Lijuan. Transition from IPv4 to IPv6 based on socket applications[C]//Proc of international conference on networking and digital society. [s. l.]:[s. n.], 2009:40-43.

[11] 蒋文娟,卢朝晖,刘家宁.基于IPv6的组播编程实例剖析[J].海南师范大学学报(自然科学版),2007,20(4):321-325.

[12] Stevens W, Fenner B, Rudoff A. UNIX网络编程第1卷:套接口API[M].杨继张,译.第3版.北京:清华大学出版社,2006.

(上接第193页)

OpenStackTutorialIEEECloudCom.pdf.

[6] Ma Yanqing, Wang Hongbo, Dong Jiankang, et al. ME2: efficient live migration of virtual machine with memory exploration and encoding[C]//Proc of 2012 IEEE international conference on cluster computing. [s. l.]:[s. n.],2012:610-613.

[7] Horvath T, Abdelzaher T F, Skadron K, et al. Dynamic voltage scaling in multitier Web servers with end-to-end delay control[J]. IEEE transactions on computers,2007,56(4):444-458.

[8] Nathuji R, Schwan K. VirtualPower: coordinated power management in virtualized enterprise systems[C]//Proceedings of twenty-first ACM SIGOPS symposium on operating systems principles. [s. l.]:[s. n.],2007:265-278.

[9] Liu L, Wang H, Liu X, et al. GreenCloud: a new architecture for green data center[C]//Proceedings of the 6th IEEE international conference on autonomic computing and communications. [s. l.]:[s. n.],2009:29-38.

[10] Imada T, Sato M, Kimura H. Power and QoS performance characteristics of virtualized servers[C]//Proc of IEEE/ACM international conference on grid computing. [s. l.]:[s. n.], 2009:232-240.

[11] RRD TOOL[EB/OL]. 2012. http://www.mrtg.org/rrdtool/index.en.html.

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
52. 基于 USB30 的设备自定义请求实现方法
53. IEEE1588 协议在网络测控系统中的应用
54. USB30 物理层中弹性缓冲的设计与实现
55. USB30 的高速信息传输瓶颈研究
56. 基于 IPv6 的 UDP 通信的实现

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
33. 基于 VxWorks 实时控制系统中文交互界面开发平台
34. 基于 VxWorks 操作系统的 WindML 图形操控界面实现方法
35. 基于 GPU FPGA 芯片原型的 VxWorks 下驱动软件开发
36. VxWorks 下的多串口卡设计
37. VxWorks 内存管理机制的研究
38. T9 输入法在 Tilcon 下的实现
39. 基于 VxWorks 的 WindML 图形界面开发方法

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
40. 基于嵌入式 Linux 下的 USB30 驱动程序开发方法研究
41. Android 手机应用开发之音乐资源播放器
42. Linux 下以太网的 IPv6 隧道技术的实现
43. Research and design of mobile learning platform based on Android
44. 基于 linux 和 Qt 的串口通信调试器调的设计及应用
45. 在 Linux 平台上基于 QT 的动态图像采集系统的设计
46. 基于 Android 平台的医护查房系统的研究与设计

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
21. DCOM 协议在网络冗余环境下的应用
22. Windows XP Embedded 在变电站通信管理机中的应用
23. XPE 在多功能显控台上的开发与应用
24. 基于 Windows XP Embedded 的 LKJ2000 仿真系统设计与实现

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
20. 基于 PowerPC 的模拟量输入接口扩展
21. 基于 PowerPC 的车载通信系统设计
22. 基于 PowerPC 的嵌入式系统中通用 IO 口的扩展方法

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
28. 基于 ARM11 的嵌入式无线视频终端的设计
29. 基于 S3C6410 的 Uboot 分析与移植
30. 基于 ARM 嵌入式系统的高保真无损音乐播放器设计
31. UBoot 在 Mini6410 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
25. 基于龙芯 2F 架构的 PMON 分析与优化
26. CPU 与 GPU 之间接口电路的设计与实现
27. 基于龙芯 1A 平台的 PMON 源码编译和启动分析
28. 基于 PC104 工控机的嵌入式直流监控装置的设计
29. GPGPU 技术研究与发展
30. GPU 实现的高速 FIR 数字滤波算法

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
7. 数据结构考题 - 第 1 章 绪论
8. 数据结构考题 - 第 2 章 线性表
9. 数据结构考题 - 第 2 章 线性表 - 答案