嵌入式Linux在物理隔离器中的应用

刘建成,李永刚
(首都师范大学 信息工程学院,北京 100051)

摘 要:传统的网络隔离方案采用的是逻辑隔离。为了达到从物理上隔离的目的,以保证更可靠的网络安全,同时保证快速的网络传输,文中提出了物理隔离器的设计方案。双Linux系统通过双端口RAM交互网络数据,并采用CPLD逻辑控制器同步双端系统,实现物理隔离和数据传输,同时在Linux内核中加入防火墙过滤,增强安全性。作为一个典型的嵌入式Linux在设备中的应用,文中阐述了开发环境建立、调试环境建立、网卡驱动实现等方面,有助于了解嵌入式Linux的开发过程和原理。

关键词:物理隔离器;Linux;双端口RAM;嵌入式
中图分类号:TP393.08 **文献标识码**:A **文章编号**:1673-629X(2006)07-0183-03

Embedded Linux's Application on Physical Isolator

LIU Jian-cheng, LI Yong-gang
(School of Information Engineering, Capital Normal University, Beijing 100051, China)

Abstract: Traditional network isolate resolution adopts logical isolating mode in order to isolate on physical layer, make the network more secure and faster networking transport. This article provides the physical isolator's design resolution. Bi-system communicates by the DPRAM, adopting one CPLD to synchronize the two systems to implement the physical isolating and data communication. In addition, adds the firewall filter in the Linux kernel to improve the security. As the typical application of embedded Linux on device, expound the development environment building, debug environment building and network card driver implementation etc. Help to understand the development steps and theory of embedded Linux.

Key words: physical isolator; Linux; DPRAM; embedded

0 引 言

网络安全是信息安全领域一个非常重要的方面,随着计算机网络的广泛应用,网络安全的重要性也日渐突出,网络安全也已经成为国家、国防及国民经济的重要组成部分。物理隔离器提供了一种很新颖的网络安全解决方式。伴随着Linux在网络设备中的广泛应用,其低廉、高效的应用性能,以及丰富的源码支持等,采用嵌入式Linux搭建网络平台渐渐成为一种趋势。

1 物理隔离器原理

1.1 物理隔离器思路

物理隔离的思路,源于两台完全不相连的计算机,使用者通过隔离控制器从一台计算机向另一台计算机拷贝数据,有时候大家形象地称为"数据摆渡"。同时在数据传输的过程中通过更加严密的安全措施增加数据传输的安全性,从而大大减少基于网络的攻击威胁。

1.2 物理隔离器实现方案

在本方案中,物理隔离器采用两个独立的8250系统,通过互斥访问共享的DPRAM,达到同一时刻两边系统相互隔离的目的。CPLD完成协调两个系统对DPRAM的访问,以保证同一个时刻只有一个系统可以访问DPRAM。DPRAM数据的读写由虚拟网卡驱动完成。图1为物理隔离器逻辑图。

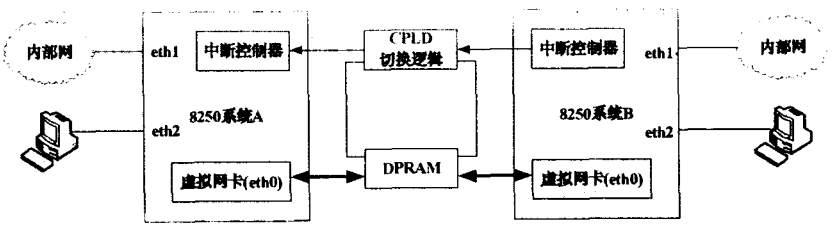

图1 物理隔离器逻辑图

在图1中,内部网和外部网通过物理隔离器交互网络数据包,控制台/管理员通过独立的PC机配置或者修改相连系统的参数,包括IP地址、防火墙过滤规则等。8250系统A和8250系统B是对等的,两边的操作系统是完全一样的,差别仅仅是具体的网络配置参数不同,以及连接的网络环境不同。以8250系统A为例,如果内网的数据

收稿日期:2005-10-30
作者简介:刘建成(1977-),男,山东莱芜人,硕士研究生,研究方向为实时操作系统、嵌入式软件环境、电子技术应用;李永刚,副教授,主要从事通信与信息系统教学。

要透过物理隔离器访问外部网,需要系统进行数据转发。在系统A有权写DPRAM的时刻,把网络数据写到DPRAM中,等CPLD切换后,系统B会把DPRAM的数据取出,转发到eth1,最后传送到外网的机器上。在系统A中,网络包的转发,可以通过网桥来实现,在eth0和eth1之间建立网桥工作模式:eth1在收到内网数据后,会直接交给eth0,eth0虚拟网卡驱动会自动选择合适的时机写入DPRAM;同样,在eth0读到DPRAM的数据后,会通过网桥,转发给eth1,如此,完成内网和外网的数据传送。为了保证数据传送的安全性,本方案采用了Linux自带的netfilter功能,通过iptables配置过滤规则,限制用户的访问,其中包括:MAC过滤、IP地址过滤、端口过滤、协议过滤等。

2 嵌入式Linux在物理隔离器系统中的应用

嵌入式Linux在网络设备中得到了广泛的应用,我们可以在网上免费获取内核源程序、软件开发环境等。

2.1 嵌入式Linux开发环境建立

2.1.1 开发编译环境建立

嵌入式Linux程序的开发需要一套交叉编译环境,将在X86上的C程序编译为PPC8250系统可以是别的目标代码。针对于PowerPC,德国的denx软件中心提供了一套完整的Linux移植开发编译环境套件:ELDK,即Embedded Linux Development Kit。

安装ELDK很简单,其过程如下:

下载并解压缩到目录:/opt/eldk-ppc-linux

进入目录:cd/opt/eldk-ppc-linux

删除如下文件:RPMS/rpm-4.0.3-1.03b_2..i386.rpm

RPMS/rpm-build-4.0.3-1.03b_2.i386.rpm

RPMS/rpm-devel-4.0.3-1.03b_2.i386.rpm

tools/usr/lib/rpm/rpmpopt-4.0.3

开始安装:/installd/opt/work ppc_82xx

到这里,开发编译环境的安装基本完成了,现在要设置一下系统环境,使eldk正常工作。在主机的Linux用户目录,编辑文件.bash_profile,加入如下内容:

```
PATH = $PATH: $HOME/bin:/opt/work/usr/bin:/opt/work/bin:/usr/bin
CROSS_COMPILE=ppc_82xx-
export PATH CROSS_COMPILE
```

这个时候,重新登陆,使该配置文件生效,开发编译环境的设置就完成了[1]。

2.1.2 下载调试环境建立

典型的嵌入式开发过程中,一般采用网络传输内核映像文件、ramdisk文件,以及其他应用程序等,这样可以保证数据传送的速度。应用程序的调试或者debug信息的输出一般用串口连接传送。如图2所示,即典型的嵌入式开发模式。

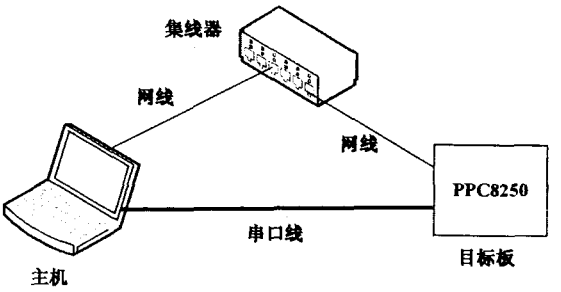

图2 开发调试连接图

内核映像文件和ramdisk文件的传送,通常用tftp协议传输,PPC平台上,一般采用PPCBoot作为bootloader程序,在PPCBoot中支持了tftp协议,用户可以指定tftp server的IP地址,利用PPCBoot的相关命令实现文件下载。对此不作详细介绍。

串口的连接,在PC主机上,可以用minicom串口工具,很方便地实现与目标平台的连接[1]。

2.2 虚拟网卡驱动实现

虚拟网卡驱动是物理隔离器软件系统中非常关键的一部分,采用标准的Linux网络设备驱动架构,负责操作系统与DPRAM的数据传递。

2.2.1 Linux网络设备驱动架构

所有的Linux网络驱动程序遵循通用的接口。一个设备就是一个device结构,它内部有自己的数据和方法。每一个设备的方法被调用时的第一个参数都是这个设备对象本身。如此,这个方法就可以存取自身的数据。一个网络设备最基本的方法有初始化、发送和接收。

初始化程序完成硬件的初始化、device中变量的初始化和系统资源的申请。发送程序dev_queue_xmit()是在驱动程序的上层协议层有数据要发送时自动调用的。一般驱动程序中不对发送数据进行缓存,而是直接使用硬件的发送功能把数据发送出去。接收数据一般是通过硬件中断来通知的。在中断处理程序里,把硬件帧信息填入一个skbuff结构中,然后调用netif_rx()传递给上层处理[2,3]。

2.2.2 虚拟网卡驱动

(1)模块初始化[4]。

模块初始化主要完成网卡设备数据结构的初始化、DPRAM的地址映射、超时定时器初始化、分配内存块等操作。

```
static int_init switch_init_module (void)
{
//DPRAM 地址映射
immap = (immap_t *) IMAP_ADDR;/* and to internal registers */
io= &immap->im_ioport;
.....
//硬件定时器初始化
io->iop_pparc &= SW_PC_PIN1_PAR;
io->iop_pdirc |= SW_PC_PIN1_DIR;
```

```
……
//分配内存
    sw_ priv = kmalloc(sizeof(struct switch_ enet_ private),
GFP_KERNEL);
    //网卡设备数据结构初始化
    dev=init_etherdev(0,0);
    ……
    sw_priv→dev=dev;
    dev→priv=sw_priv;
    dev→irq=SW_IRQ;
    dev→open=switch_enet_open;
    dev→stop=switch_enet_close;
    dev→hard_start_xmit=switch_enet_start_xmit;
    ……
    }
```

(2)设备打开操作。

打开操作完成请求注册切换中断 SW_IRQ,启动定时器等操作。

PPC8250 中请求中断,采用下面的方式:

```
result=request_8xxirq(dev→irq,switch_enet_interrupt,0,dev→
name,dev)
```

(3)发送操作。

发送过程跟普通的网卡驱动一样,只需要把待发送的数据添加到发送队列中即可。

(4)切换中断服务程序。

切换中断服务程序完成 DPRAM 的读写操作。每一次中断来临后,会读取 DPRAM 的所有数据,然后调用 netif_rx 传送到网络协议的上层;同时,读取所有待发送队列中的数据,将数据写到 DPRAM 中。

3　网络环境配置以及安全设置

要完成内外网的数据传送,需要通过正确配置网桥工作方式;同时,加入各种网络安全措施,比如设置 netfilter 过滤规则可以显著提高物理隔离器的安全性。

3.1　网桥设置

在命令行状态下,输入下列命令,配置网桥[5]:

```
#brctl addbr br0
#brctl addif br0 eth0
#brctl addif br0 eth1
#ifconfig eth0 down
#ifconfig eth1 down
#ifconfig eth0 0.0.0.0 up
#ifconfig eth1 0.0.0.0 up
#ifconfig br0 192.168.0.2 up
```

3.2　过滤规则设置

如果内核中 netfilter 功能被正确启用后,用户通过命令行或者其他方式输入的过滤规则,会被系统匹配,以决定是否允许数据包通过。命令行方式下的规则设置使用 iptables。

4　结束语

文中从整体上描述了物理隔离器的实现方案和原理,并阐述了开发编译环境建立、网卡驱动实现等。随着网络安全的重要性进一步加强,物理隔离器将得到更加广泛的应用。

参考文献:

[1] Yaghmour K. Building Embedded Linux Systems[M]. USA: O'Reilly,2003.54-64.

[2] Rubini A,Corbet J. Linux Device Driver(2nd Edition)[M]. USA:O'Reilly,2001.425-469.

[3] 毛德操,胡希明. Linux 内核源代码情景分析[M]. 杭州:浙江大学出版社,2002.119-559.

[4] 陈莉君. 深入分析 Linux 内核源代码[M]. 北京:人民邮电出版社,2002.210-325.

[5] Dyson P,Kelly-Bootle S. UNIX 大全[M]. 北京:电子工业出版社,2000.51-62.

(上接第 128 页)

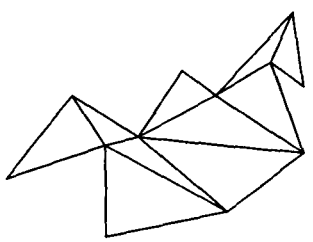

图 6　最终结果

中,由于在输出一个三角形后多边形顶点中归类受到影响的仅有有限的两个,相对于文献[5]中的算法,本算法避免了大量的不必要的运算,同时又设计了巧妙的数据结构来存放不同类别顶点的信息,因此算法效率大大提高,时间复杂度为 $O(mn)$。

参考文献:

[1] Chazelle B. Triangulation a simple polygon in linear time[R]. Technical Report CS-TR-264-90,Dept. of Computer Science ,Princeton University,1990.

[2] Preparata F P,Shamos M I. Computational Geometry[M]. New York:Springer-Verlag,1985.

[3] Kong X S, Everett H,Toussaint G T. The Graham Scan Triangulates Simple Polygons[J]. Pattern Recognition Letters, 1990,11:713-716.

[4] 杨　杰. 基于凹凸顶点判定单多边形的三角剖分[J]. 小型微型计算机系统,2000,21(9):974-975.

[5] 马小虎. 基于凹凸顶点判定的简单多边形 Delaunay 三角剖分[J]. 计算机辅助设计与图形学学报,1999,11(1):1-3.