VxWorks 网络协议栈的 MUX 接口

■ 中国矿业大学　张晓华　李智涛　徐钊

摘　要　嵌入式系统网络协议栈的灵活性在应用上是一个重要指标。介绍嵌入式实时操作系统 VxWorks 网络协议栈的MUX(Mutiplexer)接口及其使用方法。

关键词　VxWorks　网络协议栈　MUX

VxWorks是美国 Wind River System 公司（风河公司）推出的一个运行在目标机上的高性能、可裁减的嵌入式实时操作系统。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通信、军事演习、弹道制导、飞机导航等。VxWorks 操作系统包括进程管理、存储管理、设备管理、文件系统管理、网络协议及系统应用等几个部分。VxWorks 只占用很小的存储空间，并可高度裁减，保证了系统能以较高的效率运行。它可以根据用户需求进行组合，其开放式结构和对工业标准的支持使开发者只须做最少的工作即可设计有效的适合于不同用户的要求。

1　VxWorks 的网络协议栈和 MUX 接口

VxWorks 中的网络协议栈叫作 SENS（Scalable Enhanced Network Stack），即可裁减增强网络协议栈。SENS 是基于 4.4BSD TCP/IP 协议栈发展而来的，包含了许多 4.4BSD TCP/IP 协议栈没有的协议；而且 SENS 在实现一些协议功能时增加了许多新特性，如在IP 协议实现时增加了多播功能。SENS 协议栈层次如图 1 所示。

SENS 的基本特征和传统的 TCP/IP 网络协议栈相似，但从图 1 中可以看出 SENS 最大的特点是在数据链路层和网络协议层之间多了 MUX 层。在 SENS 中，网络接口的驱动程序是叫作 END（Enhanced Network Driver），即增强型网络驱动程序，它处于数据链路层。IP 层和 TCP/UDP 层合称为网络协议层。在数据链路层和网络协议层之间有应用程序接口（API），这个接口在 SENS 中叫作MUX（Multiplexer）接口。MUX 接口如图 2 所示。

在网络协议层，VxWorks 典型地使用 TCP/IP 协议（也支持其它协议）；在数据链路层典型地使用

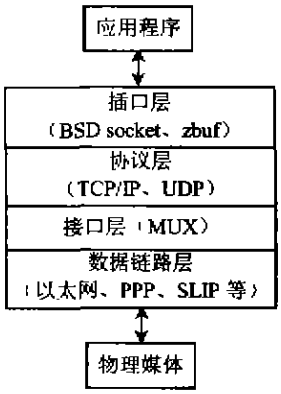

图 1　SENS 网络协议栈

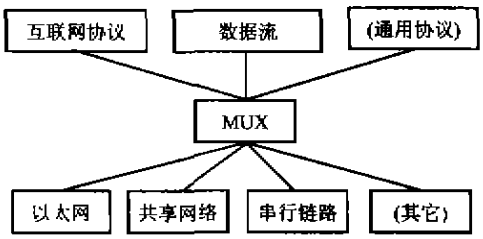

图 2　SENS 中的 MUX 接口

Ethernet，也支持其它数据传输的物理媒体，例如远距离连接使用的串行线路接入方式，如 PPP 等。但是，无论使用什么物理媒体，网络接口驱动都要用到 MUX 去与网络协议层通信（数据链路层是一个抽象概念，网络接口驱动程序则是这种抽象概念所描述的功能实现的代码）。

在 4.3BSD 中，VxWorks 的网络接口驱动和协议是紧密结合在一起的，它们通过传递特定的数据结构相互通信；而在 MUX 基础上，它们只是通过 MUX 间接地相互作用。例如，在收到一个包后，网络接口驱动并没有直接与协议层连接。同样地，当网络接口驱动准备好向协议层发送数据时，驱动程序会调用一个 MUX 提供的功能（函数）。这个功能（函数）具体负责将数据传给协议层的动作细节。应用

MUX 的主要目的是把网络接口驱动和协议层分开，这样就使得网络接口驱动和协议层彼此基本上保持独立。这种独立性使得加载一个新的协议或网络接口驱动变得非常容易。例如：如果要加一个新的网络接口驱动，所有现有的基于 MUX 的协议就都可以用这个新的网络接口驱动程序；同样，如果要加一个新的基于 MUX 的协议，现有的网络接口驱动也能够用 MUX 来与新协议通信。

2 MUX 接口工作流程分析

MUX 层作为独立的一个网络层有其自己的功能函数，但这些功能函数只是其上下两层通信的接口。网络协议层和网络驱动与 MUX 接口的调用关系如图 3 所示。

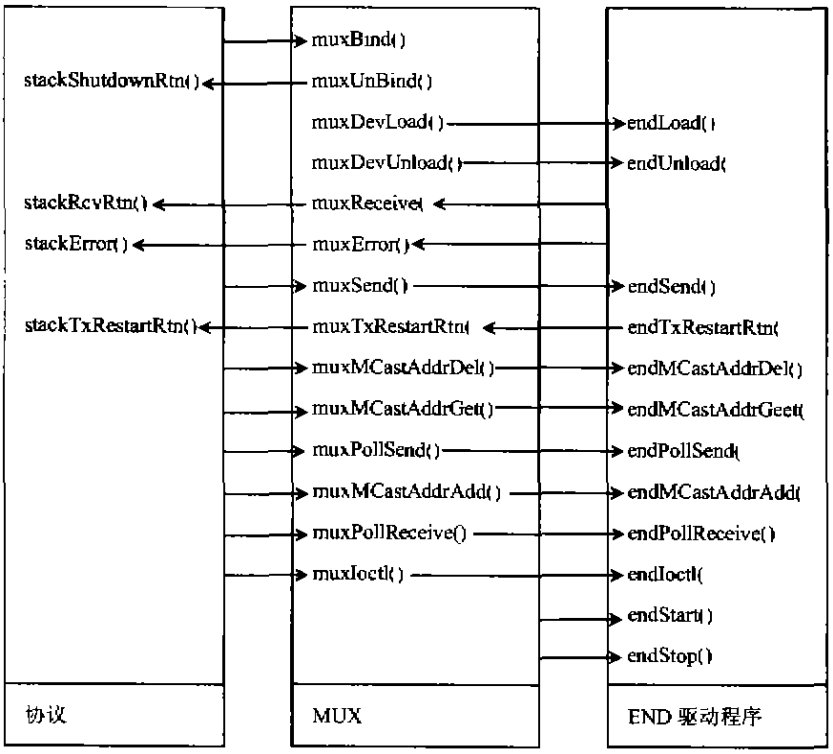

图 3　网络协议层、MUX、网络驱动

网络协议提供下面的接口功能函数：

① stackShutdownRtn( )

② stackError( )

③ stackRcvRtn( )

④ stackTxRestartRtn( )

当 MUX 接口层需要与协议层相互通信时，就调用以上的功能函数。想要使网络协议层能够使用 MUX，必须至少实现以上四个功能函数。

MUX 则实现 muxBind( )、muxUnBind( )、muxDevLoad( )等等。网络协议层和网络驱动接口都要根据各自的需要使用 MUX 接入点。由于 MUX 是由系统提供的，不需要在应用时再进行额外的编码工作；只要在使用时，填入正确的参数即可。

例如在 VxWorks 中，muxDevLoad 是这样定义的：

```
END_OBJ* muxDevLoad
  (
  int unit,                                  /* 设备号码 */
  END_OBJ*(*endLoad)(char*, void*),/*调用设备函数 */
  char* pInitString,                         /* 初始化字符串 */
  BOOL loaning,                              /* 存储标识 */
  void* pBSP                                 /*调用BSP功能的函数*/
  )
```

其它功能函数在 muxLib.h 文件中有详细定义。

网络接口的驱动程序要完成 endLoad()、endUnload( )、endSend( )等功能函数。MUX 使用这些功能函数来与网络驱动程序通信。当编写或加载一个使用 MUX 的网络驱动程序时，必须实现图 3 中 END 的所有功能。这些功能函数都是针对具体的网络接口，即每一个网络驱动程序中都要有这些功能函数。

3 MUX 的应用

3.1 系统如何通过 MUX 启动和使用网络接口驱动程序——END

系统启动时要通过任务（与进程概念相近，是嵌入式操作系统中的执行单元）来执行以下功能：

① 从存储器中装载并启动 END；

② 注册用来处理 END 的中断；

③ 通过 END 来实现处理包的功能。

在系统启动时，VxWorks 产生一个 tUsrRoot 任务来执行以下的功能：首先初始化网络任务的工作队列，然后产生一个 tNetTask 来处理网络任务工作队列中的任务。

tNetTask 任务调用 muxDevLoad( )来装载网络接口驱动，在 tNetTask 中已经定义了网络驱动设备的 endLoad( )接入点，muxDevLoad( )则也要执行 endLoad( )。endLoad( )执行设备初始化并且返回一个名为 END_OBJ 的结构。MUX 通过在 END_OBJ 上加一个指针，指向能完成将数据包向 MUX 上层发

送的功能（函数）。然后MUX把返回的END_OBJ加到END_OBJ结构链表中。这个链表包括目前系统中所有可用的网络设备。当从muxDevLoad( )返回后，网络设备就准备好可以使用了。

我们必须调用sysIntConnect()来注册网络接口设备的中断处理。最典型的调用sysIntConnect()是在网络接口设备的endStart()中。当通过muxDevLoad( )来装载网络接口设备时，就会调用muxDevStart( )来启动该设备。muxDevStart( )就会调用endStart( )，从而进行中断处理的注册。

系统启动后，就要依靠中断来使用该设备。当从网络设备的中断来到时，VxWorks激活该设备驱动程序所注册的中断服务。中断服务的工作量应该是最小的，只需完成从本地硬件上取到包即可。为了使中断的锁定时间最少，中断服务应该仅处理那些要求最少执行时间的任务，例如出错和状态改变。中断服务为了让所有耗时的工作在任务级别处理，应该将其排队。例如：要使在任务级别处理包接受的工作排队，中断服务必须调用netJobAdd( )。在输入的时候，这个例行程序（Routine）收到一个功能函数的指针并且直到收到五个额外的参数（指针所指功能函数的参数）。

```
STATUS netJobAdd
(
FUNCPTR routine.  /*在工作程序队列中要加的例行程序 */
int     param1.   /* 这个例行程序的第一个参数 */
int     param2.   /*这个例行程序的第二个参数 */
int     param3.   /*这个例行程序的第三个参数 */
int     param4.   /*这个例行程序的第四个参数 */
int     param5.   /*这个例行程序的第五个参数 */
)
```

如果调用netJobAdd()，就必须定义网络驱动在任务级别处理包的接入点。NetJobAdd例行程序将功能函数调用（包括其参数）放入tNetTask的任务队列中。VxWorks使用tNetTask处理任务级别的网络处理功能。

这里只是举例说明了接收包的情况，其它情况下netJobAdd( )也一样能执行对应的入列功能。

3.2 基于MUX的网络协议层和网络接口驱动程序

基于MUX的特点：提供一个接口，使与其相连各层程序的编写只需在其基础上编写即可。可以说不论是网络协议层还是网络接口驱动程序都可以把MUX看作应用程序接口(API)。在VxWorks目标系统中加载一个网络接口设备的驱动程序就和编写一个应用程序一样的简单，具体步骤如下：

① 编译驱动程序的源代码并在VxWorks镜像中加载；

② 编辑target/src/config/BSP/configNet.h；

③ 编辑BSP的config.h文件。

注意，如果不重新编译新的boot ROMs(启动ROM)，那么就不能使用新的END。这就是说，只能启动一个没有END的目标系统，所以必须编辑配置文件，才能使用新的END。

由图3可知，基于MUX的网络协议与MUX向上绑定，而网络接口驱动是与MUX向下绑定的。协议层的主要功能是对传输层和应用程序提供接口。协议层的代码编写也是通过MUX接口提供的接口进行编程，这和通用计算机有很大的相似之处，这里就不再赘述。

VxWorks的网络协议栈SENS中的MUX接口对于开发应用是十分方便的。利用这一特点可以在不同的网络环境中使用嵌入式操作系统。

参考文献

1 SENS for Tornado. Wind River System, Inc.

2 Richard Stevens W. The Protocols. TCP/IP Illustrate,Vol 1

3 Wright Gary R, Richard Stevens W. The Implementation. TCP/IP Illustrate, Vol 2

（上接第21页）这些都必须重新考虑。许多PBO进程运行于多处理器上，它们之间协同工作要进行的网格计算也不容易。在PBO组件数目较多时，数据复制机制能否具有高可靠性和及时性也都是可能遇到的问题。

参考文献

1 Stewart D B, Volpe R A, Khosla P K. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. IEEE trans. on software engineering, 1997,23(12)

2 乔颖，王宏安，戴国忠. 实时系统开发方法研究. 计算机科学，2001,28(2)

3 万琳，宫云战. 一种嵌入式控制软件的可靠性测试. 计算机工程，2001,27(3)

4 张升昕，董淳. 对系统动态更新的探讨与设计. 计算机科学，1999,26(2)

5 蔡持峰，等. 实时控制系统需求描述方法及其应用. 计算机科学，1999,28(3)