uIP 中 UDP 协议实现的改进*

曹欲晓,韩 磊

(南京工程学院 计算机工程学院,江苏 南京 211167)

摘 要:uIP 作为一种广泛使用的轻量级嵌入式 TCP/IP 协议栈,其 UDP 协议的实现还不够完善,目前最新的 1.0 版本中仅实现了 UDP 客户端,尚没有实现 UDP 服务端。为此,对其进行了以下三方面的改进:UDP 服务端口的初始化;接收到 UDP 客户端数据包后的端口号判断及匹配;UDP 服务端发送报文后目的端口的释放。经过以上改进后,实验证明,uIP 1.0 中的 UDP 实现了服务端的功能。

关键词:uIP;嵌入式 TCP/IP 协议栈;UDP;端口

中图分类号:TP393.04 **文献标识码**:A **文章编号**:1674-7720(2010)21-0052-03

Improvement of UDP implementation in uIP

CAO Yu, Xiao, HAN Lei

(School of Computer Engineering, Nanjing Institute of Technology, Nanjing 211167, China)

Abstract: uIP is a kind of widely used lightweight embedded TCP/IP protocol stack, but the implementation of UDP protocol of uIP is not perfect. In uIP's newest version uIP 1.0, only UDP client has been implemented, and UDP server hasn't been complemented so far. In this paper, three points of improvement on uIP 1.0 are given. Firstly, initialize UDP server port. Secondly, decise and match the port of UDP data packet from UDP client. At last, release the destination port of UDP server after data packet has been sent. Laboratory result showed that uIP 1.0 can run as UDP server rightly after it is improved.

Key words: uIP; embedded TCP/IP protocol stack; UDP; port

随着嵌入式技术、网络技术的发展,实现网络互联已经成为嵌入式系统发展的一个必然趋势。在目前的技术条件下,越来越多的嵌入式系统选择了 TCP/IP 作为与其他计算机系统互联的网络协议。嵌入式 TCP/IP 协议栈已经成为嵌入式系统研究与应用中的一个重要领域。

由于嵌入式系统的软硬件资源都较为有限,大多数嵌入式系统中运行的 TCP/IP 协议栈均根据嵌入式系统的特点进行了相应的裁剪。目前应用比较广泛的嵌入式 TCP/IP 协议栈有:ucTCP-IP、LWIP、uIP、Linux TCP/IP 等。其中 uIP 是专为 8 bit 和 16 bit 的嵌入式微控制器设计的微型 TCP/IP 协议栈,它具有良好的互操作性,并遵循 RFC 标准。uIP 协议栈的特点是具有很小的代码量,运行时需要的内存很少,实现了常用的 TCP/IP 协议;代码注释详尽,可以用于商业或非商业用途[1]。由于具有上述特点,uIP 被广泛应用在嵌入式系统的网络互联中。

1 uIP 协议栈的体系结构

在使用 uIP 的嵌入式系统的软件体系结构中,uIP 协议栈相当于一个代码库,它通过一系列的函数实现与底层硬件和上层应用程序的通信。uIP 协议栈与系统底层和上层应用之间的关系如图 1 所示[2]。

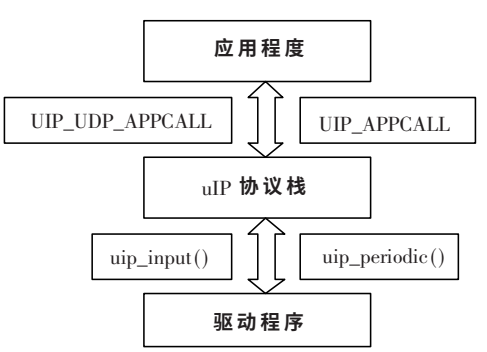

图 1 uIP 协议栈体系结构

*基金项目:南京工程学院科研基金项目(KXJ08070)

从图 1 可以看出，uIP 协议栈主要提供了 *uip_input*() 和 *uip_periodic*()2 个函数供系统底层调用。uIP 协议栈与应用程序的主要接口是 *UIP_APPCALL*() 和 *UIP_UDP_APPCALL*()。

uIP 初始化时调用 *uip_init*() 函数，它的主要功能是初始化协议栈的监听端口，并把所有连接设置为关闭状态。当网络控制芯片驱动程序接收到一个数据包时，驱动程序将数据包放入全局缓冲区 uip_buf 中，同时把包的大小赋给全局变量 *uip_len*。然后 uIP 的主控部分调用 *uip_input*() 函数，该函数将会根据数据包首部的协议标识处理这个包，并在需要时调用上层应用程序。当 *uip_input*() 返回时，一个输出数据包被放在同一个全局缓冲区 uip_buf 中，其大小赋给 *uip_len*。如果 *uip_len* 是 0，则说明没有包要发送，否则主控部分调用底层系统的发包函数将数据包发送到网络上[3]。

uIP 周期计时用于驱动所有的 uIP 内部时钟事件。当周期计时激发后，每一个 TCP 连接都会调用 uIP 函数 *uip_periodic*()。类似于 *uip_input*() 函数，*uip_periodic*() 函数返回时，输出的 IP 包要放到 uip_buf 中，供底层系统查询 *uip_len* 的大小以决定是否发送。

由于使用 TCP/IP 的应用很多，因此应用程序作为单独的模块由用户实现。uIP 提供一系列接口供用户程序调用，其中大部分接口是作为 C 的宏命令出现的，之所以这样做主要是考虑到速度、代码大小、效率和堆栈的使用。用户需要把对网络数据包的处理函数作为接口提供给 uIP，并将这个函数定义为宏 *UIP_APPCALL*() 或者 *UIP_UDP_APPCALL*()。*UIP_APPCALL*() 是用户对 TCP 数据包的处理，*UIP_UDP_APPCALL*() 是用户对 UDP 数据包的处理[4]。这样，uIP 在接收到底层传来的数据包后，在需要送到上层应用程序处理的地方，直接调用 *UIP_APPCALL*() 或者 *UIP_UDP_APPCALL*() 即可，无需修改 uIP。

2 uIP 的 UDP 协议分析

2.1 UDP 协议的实现

当 uIP 接收到一个 UDP 数据包后，首先从包头中取出数据的长度，然后重新对包进行校验，如果校验和不对，则直接丢掉这个包。如果校验无误，则对收到的包进行解复用。此时进行如下判断：

```
if(uip_udp_conn->lport != 0 &&
    UDPBUF->destport == uip_udp_conn->lport &&
    (uip_udp_conn->rport == 0 ||
    UDPBUF->srcport == uip_udp_conn->rport) &&
    (uip_ipaddr_cmp(uip_udp_conn->ripaddr, all_zeroes_addr) ||
uip_ipaddr_cmp (uip_udp_conn ->ripaddr, all_ones_addr) ||
uip_ipaddr_cmp (BUF ->srcipaddr, uip_udp_conn ->ripaddr)))
```

上述代码中用到的主要变量、数据结构和函数的含义是：

uip_udp_conn->lport：本地 UDP 的源端口；

uip_udp_conn->rport：本地 UDP 的目的端口；

UDPBUF->srcport：接收到的数据包中的源端口；

UDPBUF->destport：接收到的数据包中的目的端口；

uip_udp_conn->ripaddr：本地 UDP 的目的 IP 地址；

BUF->srcipaddr：接收到的数据包中的源 IP 地址；

all_zeroes_addr：IP 地址 0.0.0.0；

all_ones_addr：IP 地址 255.255.255.255；

uip_ipaddr_cmp：IP 地址比较函数，如果参加比较的两个 IP 地址相等，则返回 1。

在 uIP 的实现中，如果以上判断语句为真，则对接收到的数据包进行处理，处理过程包括调用用户上层处理程序 *UIP_UDP_APPCALL*()、构造新包的包头、计算新包的校验和等，然后将构造好的返回 UDP 包送到 IP 层进行处理。

2.2 UDP 实现的不足

通过对 uIP 中 UDP 协议实现过程的分析可以发现，uIP 没有提供初始化指定端口的函数，仅提供了一个对给定 IP 地址上给定端口建立 UDP 连接的函数，其原型是 struct uip_udp_conn*uip_udp_new(uip_ipaddr_t*ripaddr,u16_t rport)。由于作为服务端运行时必须指定监听端口[5]，而 uIP 没有提供此功能，因此要让 uIP 作为服务端运行，必须对 uIP 进行改进。

3 uIP 中 UDP 协议的改进

3.1 增加初始化 UDP 服务端口

UDP 协议作为服务端运行时，同 TCP 一样，必须在某个指定端口上监听客户端是否有数据包发送，如果有则还要接收数据包，这就要求在 uIP 记录 UDP 连接的数据结构 uip_udp_conn 中设置本地端口号一项，具体实现步骤如图 2 所示。

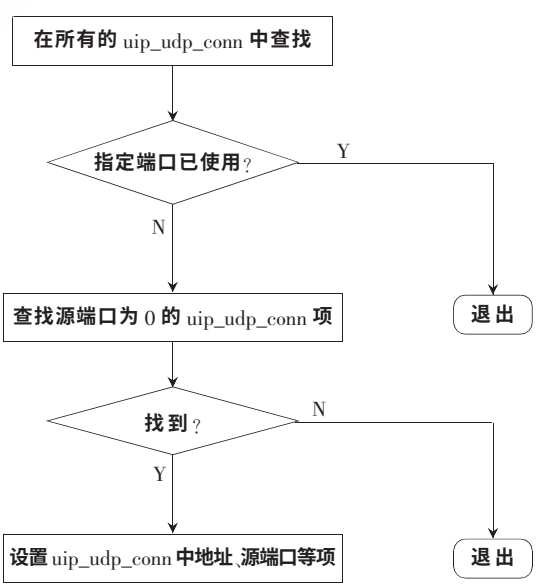

图 2 UDP 服务端口初始化

3.2 IP 地址、端口号的判断及匹配

uip_process 函数接收到网络控制芯片驱动程序送来的数据包后，当判断出收到的包是 UDP 包，执行 2.1 中的判断并且得到结果为真后，但还需要再做以下工作：如果 uip_udp_conn 中的目的端口号为 0，则说明这是一个来自客户端的首次与服务端进行通信的数据包，服务端尚不知道此客户端的源端口，因此要把 uip_udp_conn 中的目的端口号设为收到的包中的源端口号，把 uip_udp_conn 中的目的 IP 地址设为收到的包中的源 IP 地址，具体代码如下：

```
if(uip_udp_conn->rport==0)
{
uip_udp_conn->rport=UDPBUF->srcport;
memcpy(uip_udp_conn->ripaddr,UDPBUF->srcipaddr,
sizeof(uip_ipaddr_t ));
}
```

3.3 UDP 服务端目的端口的释放

UDP 服务端的端口应该可以为来自多个客户端的请求提供服务，而 UDP 本身是一种无连接的传输层协议，因此在每次 uIP 作为服务端的 UDP 通信结束之后，还要释放 uip_udp_conn 中记录的目的端口号，以便下次接收来自不同 IP、不同端口的新请求，否则当来自其他端口的请求到达时，uIP 会不予响应。

在 uIP 的官方网站上下载到 uIP 1.0 的源代码之后，按照本文给出的几个步骤对 uIP 1.0 进行改造之后，利用 gcc 编译器把 uIP 1.0 编译成 S3C2410 上的可执行代码，把基于 S3C2410 的开发板作为 UDP 服务器，运行 Windows XP 的 PC 机作为客户端，两者通过一条交叉网线相联，在 PC 机上的测试程序发出 UDP 请求后，运行在 S3C2410 上的 uIP 可以对 PC 通过 UDP 协议发出的数据进行处理，并给 PC 作出正确的回复。实验证明，通过对 uIP 进行本文所述的改进之后，uIP 具有了作为 UDP 服务端的能力。

参考文献

[1] http://www.sics.se/~adam/uip/index.php/Main_Page.

[2] ADAM D. The uIP embedded TCP/IP stack the uIP 1.0 reference manual. June 2006.

[3] ADAM D. Full TCP/IP for 8-bit architectures [C]. In Proceedings of the First International Conference on Mobile Applications, Systems and Services (MOBISYS 2003), San Francisco, May 2003.

[4] ADAM D, OLIVER S, THIEMO V, et al. Protothreads: simplifying event-driven programming of memory-constrained embedded systems [C]. In Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006), Boulder, Colorado, USA, November 2006.

[5] FOROUZAN B A, FEGAN S C 著. TCP/ZP 协议簇 [M]. 谢希仁等，译. 北京：清华大学出版社，2006.

（收稿日期：2010-07-05）

作者简介：

曹欲晓，男，1971 年生，讲师，硕士，主要研究方向：嵌入式系统，计算机网络。

韩磊，男，1982 年生，讲师，硕士，主要研究方向：嵌入式系统，模式识别。