第二章　uIP 协议栈分析

2．1　uIP 特性

uIP 由瑞典计算机科学学院(网络嵌入式系统小组)的 Adam Dunkels 开发。其源代码由 C 语言编写，并完全公开，所有代码和相关说明文档可以到 http://dunkels.com/adam/uip/下载。最新版本是 uIP1.0 版本，本书移植和使用的版本正是此版本。

uIP 协议栈去掉了完整的 TCP/IP 中不常用的功能，简化了通讯流程，但保留了网络通信必须使用的协议，设计重点放在了 IP/TCP/ICMP/UDP/ARP 这些网络层和传输层协议上，保证了其代码的通用性和结构的稳定性。

由于 uIP 协议栈专门为嵌入式系统而设计，因此还具有如下优越功能：

（1） 代码非常少，其协议栈代码不到 6K，很方便阅读和移植。

（2） 占用的内存数非常少，RAM 占用仅几百字节。

（3） 其硬件处理层、协议栈层和应用层共用一个全局缓存区，不存在数据的拷贝，且发送和接收都是依靠这个缓存区，极大的节省空间和时间。

（4） 支持多个主动连接和被动连接并发。

（5） 其源代码中提供一套实例程序：web 服务器，web 客户端，电子邮件发送程序(SMTP 客户端)，Telnet 服务器， DNS 主机名解析程序等。通用性强，移植起来基本不用修改就可以通过。

（6） 对数据的处理采用轮循机制，不需要操作系统的支持。

由于 uIP 对资源的需求少和移植容易，大部分的 8 位微控制器都使用过 uIP 协议栈, 而且很多的著名的嵌入式产品和项目(如卫星，Cisco 路由器，无线传感器网络)中都在使用 uIP 协议栈。

2．2　uIP 架构

uIP相当于一个代码库，通过一系列的函数实现与底层硬件和高层应用程序的通讯，对于整个系统来说它内部的协议组是透明的，从而增加了协议的通用性。uIP协议栈与系统底层和高层应用之间的关系如图2－1所示。

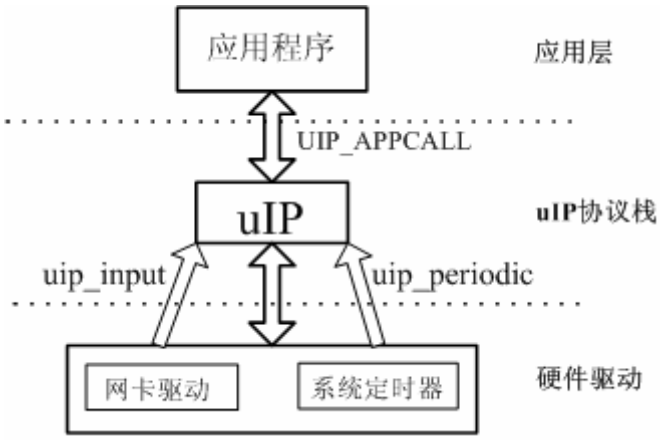

图2－1 uIP在系统中的位置

从上图可以看出，uIP 协议栈主要提供了三个函数供系统底层调用。即 uip_init(), uip_input() 和 uip_periodic() 。其与应用程序的主要接口是

UIP_APPCALL( )。

uip_init()是系统初始化时调用的，主要初始化协议栈的侦听端口和默认所有连接是关闭的。

当网卡驱动收到一个输入包时，将放入全局缓冲区 uip_buf 中，包的大小由全局变量 uip_len 约束。同时将调用 uip_input()函数，这个函数将会根据包首部的协议处理这个包和需要时调用应用程序。当 uip_input()返回时，一个输出包同样放在全局缓冲区 uip_buf 里，大小赋给 uip_len。如果 uip_len 是 0，则说明没有包要发送。否则调用底层系统的发包函数将包发送到网络上。

uIP 周期计时是用于驱动所有的 uIP 内部时钟事件。当周期计时激发，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 的大小发送。

由于使用 TCP/IP 的应用场景很多，因此应用程序作为单独的模块由用户实现。uIP 协议栈提供一系列接口函数供用户程序调用，其中大部分函数是作为 C 的宏命令实现的，主要是为了速度、代码大小、效率和堆栈的使用。用户需要将应用层入口程序作为接口提供给 uIP 协议栈，并将这个函数定义为宏 UIP_APPCALL()。这样，uIP 在接受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，调用 UIP_APPCALL( )。在不用修改协议栈的情况下可以适配不同的应用程序。

2．3　uIP 在 MCS-51 单片机上的移植

1．为此项目建立一个 keil C 工程，建立 src 目录存放源文件。

2．通过阅读 uip-1.0\unix\main.c，了解 uIP 的的主循环代码架构，并将 main.c 放到 src 目录下。

3．仿照 uip-1.0\unix\tapdev.c 写网卡驱动程序，与具体硬件相关。这一步比较费点时间，不过好在大部分网卡芯片的驱动程序都有代码借鉴或移植。驱动需要提供三个函数，以 RTL9019AS 驱动为例。

etherdev_init（）：网卡初始化函数，初始化网卡的工作模式。

u16_t etherdev_read(void)：读包函数。将网卡收到的数据放入全局缓存区 uip_buf 中，返回包的长度，赋给 uip_len。

void etherdev_send(void)：发包函数。将全局缓存区 uip_buf 里的数据（长度放在 uip_len 中）发送出去。

所以，收包和发包主要是操作 uip_buf 和 uip_len。具体驱动分析可参考《第三章　网络芯片的驱动》。

4．由于 uIP 协议栈需要使用时钟，为 TCP 和 ARP 的定时器服务。因此使用单片机的定时器 0 用作时钟，每 20ms 让计数 tick_cnt 加 1，这样，25 次计数（0.5S）满了后可以调用 TCP 的定时处理程序。10S 后可以调用 ARP 老化程序。对 uIP1.0 版本，增加了 timer.c/timer.h，专门用来管理时钟，都放到 src 下。

5．uIP 协议栈的主要内容在 uip-1.0\uip\下的 uip.c/uip.h 中，放到 src 下。如果需要 ARP 协议，需要将 uip_arp.c 和 uip_arp.h 也放到 src 下。

6．uipopt.h/uip-conf.h 是配置文件，用来设置本地的 IP 地址、网关地址、MAC 地址、全局缓冲区的大小、支持的最大连接数、侦听数、ARP 表大小等。需要放在 src 下，并且根据需要配置。在 V1.00 版本中对配置做了如下修改：

（1）配置 IP 地址，默认先关 IP，在初始化中再设定。

```
#define UIP_FIXEDADDR     0
#define UIP_IPADDR0       192
#define UIP_IPADDR1       168
#define UIP_IPADDR2       1
#define UIP_IPADDR3       9
#define UIP_NETMASK0      255
#define UIP_NETMASK1      255
#define UIP_NETMASK2      255
#define UIP_NETMASK3      0
#define UIP_DRIPADDR0     192
#define UIP_DRIPADDR1     168
#define UIP_DRIPADDR2     1
#define UIP_DRIPADDR3     1
```

（2）使能 MAC 地址

```
#define UIP_FIXEDETHADDR 1
#define UIP_ETHADDR0      0x00
#define UIP_ETHADDR1      0x4f
#define UIP_ETHADDR2      0x49
#define UIP_ETHADDR3      0x12
#define UIP_ETHADDR4      0x12
#define UIP_ETHADDR5      0x13
```

（3）使能 ping 功能

```
#define UIP_PINGADDRCONF   1
```

（4）关闭主动请求连接的功能

```
#define UIP_ACTIVE_OPEN 0
```

（5）将 uip_tcp_appstate_t 定位 u8_t 类型。

（6）由于单片机是大端结构，因此宏定义需要修改

```
#define UIP_CONF_BYTE_ORDER         UIP_BIG_ENDIAN
```

（7）暂时不移植打印信息，先关闭

```
#define UIP_CONF_LOGGING            0
```

（8）定义数据结构类型

```
typedef unsigned char u8_t;
typedef unsigned int u16_t;
typedef unsigned long u32_t;
```

7. 如果使用 keil C 的小模式编译，需要在大部分的 RAM 的变量前增加 xdata。

8. data 为 keil C 的关键词，代码中所有出现 data 的地方（主要是参数、局部变量、结构体成员）改为 pucdata 或 ucdata。

9. 解决编译过程中的错误。uIP 协议栈为 C 语言编写，编译过程中的问题比较少，并且容易解决。

2．4 uIP 的主控制循环

通过实际的代码说明 uIP 协议栈的主控制循环。

```
void main(void)
```

```
{
    /*省略部分代码*/
    /*设置 TCP 超时处理时间和 ARP 老化时间*/
    timer_set(&periodic_timer, CLOCK_CONF_SECOND / 2);
    timer_set(&arp_timer, CLOCK_CONF_SECOND * 10);
    /*定时器初始化*/
    init_Timer();
    /*协议栈初始化*/
    uip_init();
    uip_arp_init();
    /*应用层初始化*/
    example1_init();
    /*驱动层初始化*/
    etherdev_init();
    /*IP 地址、网关、掩码设置*/
    uip_ipaddr(ipaddr, 192,168,1,9);
    uip_sethostaddr(ipaddr);
    uip_ipaddr(ipaddr, 192,168,1,16);
    uip_setdraddr(ipaddr);
    uip_ipaddr(ipaddr, 255,255,255,0);
    uip_setnetmask(ipaddr);
    /*主循环*/
    while(1)
    {

    /*从网卡读数据*/
    uip_len = etherdev_read();
    /*如果存在数据则按协议处理*/
    if(uip_len > 0)
    {
        /*收到的是 IP 数据，调用 uip_input()处理*/
        if(BUF->type == htons(UIP_ETHTYPE_IP))
        {
            uip_arp_ipin();
            uip_input();
            /*处理完成后，如果 uip_buf 中有数据，则调用 etherdev_send 发
            送出去*/
            if(uip_len > 0)
                {
                    uip_arp_out();
                    etherdev_send();
                }
        }
     /*收到的是 ARP 数据，调用 uip_arp_arpin()处理*/
```

```
        else if(BUF->type == htons(UIP_ETHTYPE_ARP)) {
      uip_arp_arpin();
           if(uip_len > 0)
             {
               etherdev_send();
           }
          }
        }
      /*查看 0.5S 是否到了，到了则调用 uip_periodic 处理 TCP 超时程序*/
        else if(timer_expired(&periodic_timer))
        {
           timer_reset(&periodic_timer);
           for(i = 0; i < UIP_CONNS; i++)
           {
             uip_periodic(i);
             if(uip_len > 0)
              {
                uip_arp_out();
                etherdev_send();
              }
              }
              /*查看 10S 是否到了，到了则调用 ARP 处理程序*/
              if(timer_expired(&arp_timer))
              {
                timer_reset(&arp_timer);
                uip_arp_timer();
              }
        }
    }
    return;
}
```

2．5　uIP 协议栈提供的主要接口

提供的接口在 uip.h 中，为了减少函数调用造成的额外支出，大部分接口函数以宏命令实现的。

1．初始化uIP协议栈：uip_init()
2．处理输入包：uip_input()
3．处理周期计时事件：uip_periodic()
4．开始监听端口：uip_listen()
5．连接到远程主机：uip_connect()
6．接收到连接请求：uip_connected()
7．主动关闭连接：uip_close()
8．连接被关闭：uip_closed()

9．发出去的数据被应答：uip_acked()
10．在当前连接发送数据：uip_send()
11．在当前连接上收到新的数据：uip_newdata()
12．告诉对方要停止连接：uip_stop()
13．连接被意外终止：uip_aborted()

版本：V1.0　初稿，欢迎指导

作者：gateway

邮箱：**gatewaytech@126.com**

ＱＱ：　1079197758

修改日期：2009.3.26