uC/OS-II 支持下的嵌入式 TCP/IP 协议应用

杨天怡 陈 禾 柴 毅
(重庆大学自动化学院 重庆 400044)

摘 要 针对采用了"前后台系统"的嵌入式 TCP/IP 协议带来数据帧传输实时性差和难以添加应用层模块的问题,通过将嵌入式实时操作系统 uC/OS-II 和已被广泛运用于 8、16 位单片机系统的 uIP 协议栈的结合,优化了 uIP 协议栈各个网络模块的运行。同时针对工控设备的网络通信特点,将 UDP 协议作为协议栈的基本数据传输方式,从而提高了 uIP 协议栈数据帧传输的实时性。实验表明,在将整合后的协议栈移植到一个嵌入式的 websever 后,系统具有较高的数据传输实时性并且运行稳定。

关键词 uC/OS-II,嵌入式系统,uIP 协议栈

Embedded TCP/IP Protocol Based on uC/OS-II

YANG Tian-Yi CHEN He CHAI Yi
(College of Automation, Chongqing University, Chongqing 400044)

Abstract An embedded Real-time OS is used to overcome the problem of bad Real-time and expansibility in commonly embedded TCP/IP which is based on "front and back system". uC/OS-II is utilized to optimize the administration of each net module in uIP protocol which had been widely used in SCM system. In allusion to the character of industry control communication, at the same time, the UDP protocol became basic communication mode for improved the real-time of system. Experiment indicates that the websever has high real-time performance and stable running environment after the advanced uIP protocol had been replanted in the system.

Keywords uC/OS-II, Embedded system, TCP/IP, uIP

1 引言

目前,随着互联网的发展,越来越多的工业测控设备已经将网络接入功能作为其默认配置,以实现设备的远程监控和信息分布式处理。为了实现工控设备的以太网接入,一个适合的 TCP/IP 协议是必需的。在高级操作系统中可以完全支持的一个完整 TCP/IP 协议族,在嵌入式系统中却很难做到,也不需要做到。针对这种情况出现了嵌入式 TCP/IP 协议栈[2]。

在商业嵌入式 TCP/IP 协议栈大都相当昂贵的情况下,很多人转而使用一些源代码公开的免费协议栈,并加以改造应用。目前较为著名的免费协议栈有:lwIP(Light weight TCP/IP Stack)和瑞典计算机科学研究所 Adam Dunkels 开发的 uIP0.9(TCP/IP stack for micro)[4]。但是这些协议在工控设备上使用时,由于缺乏操作系统的支持,采用超级循环方式运行的 TCP/IP 协议效率较低,同时增加了程序管理网络模块和应用模块的设计难度。

uC/OS-II 是一种针对微控制器的实时操作系统,对于大多 ROM、RAM 有限而实时性要求较高的系统都是一个比较理想的选择。文中选用广泛使用的 uIP0.9 协议,将其和 uC/OS-II 相结合,达到提高 uIP0.9 协议运行效率和数据帧传输的实时性。

2 嵌入式 TCP/IP 协议栈 uIP

uIP 协议栈是瑞典计算机科学研究所 Adam Dunkels 开发的。uIP 出现后就被广泛地运用到各种嵌入式系统,自从 0.6 版本以来就被移植到 MSP430、AVR 和 Z80 等多种处理器上[4]。现在已经发展到了 uIP0.9。这种嵌入式 TCP/IP 协议栈去掉了许多完整协议栈中不常用的功能,而保留网络通信所必要的协议机制。具有以下特点:完整的说明文档和公开的源代码;适用于 8/16 位单片机代码占用量和 RAM 资源要求(见表 1);高度可配置性,以适应不同资源条件和应用场合;支持 ARP、IP、ICMP、TCP、UDP(可选)等必要的功能特性;支持多个主动连接和被动连接,并支持连接的动态分配和释放;简易的应用层接口和设备驱动层接口;完善的示例程序和应用协议实现范例[6]。

表 1 uIP 的代码大小和 RAM 占用情况[4]

协议模块	代码大小/B	使用的 RAM/B
ARP	1324	118
IP/ICMP/TCP	3304	360
HTTP	994	110
校验和函数	636	0
数据包缓存	0	400
总和	6258	988

注:配置为 1 个 TCP 听端口,10 个连接,10 个 ARP 表项,400 字节数据包缓存。

3 uC/OS-II 与 uIP TCP/IP 协议栈的结合

3.1 uC/OS-II 任务管理

uC/OS-II 是一段在嵌入式系统启动后首先执行的背景程序,uC/OS-II 根据各个任务的要求,进行资源(包括存储器、外设等)管理、消息管理、任务调度、异常处理等工作。在 uC/OS-II 支持的系统中,每个任务均有一个优先级,uC/OS-II 根据各个任务的优先级,动态地切换各个任务,保证对实时

性的要求。μC/OS-Ⅱ可以管理64级任务，但系统保留了4个最高优先级和4个最低优先级。因此用户可以使用56个应用任务。必须给每个不同的任务赋予不同的优先级，μC/OS-Ⅱ总是运行进入就绪状态优先级最高的任务[2]。

在uC/OS-Ⅱ中每个任务都是一个无限的循环，每个任务都处在以下5种状态之一的状态下，这5种状态是休眠态，就绪态、运行态、挂起态（等待某一事件发生）和被中断态（参见图1）[2]。

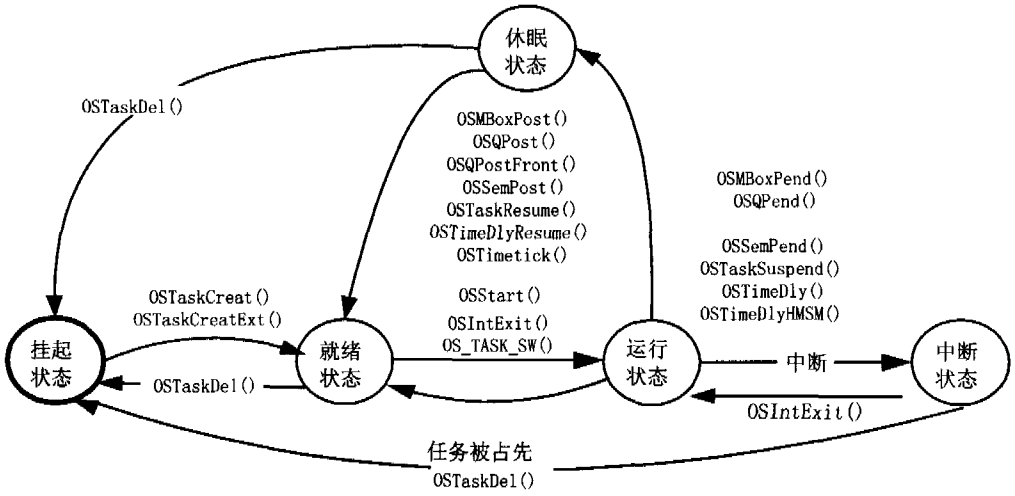

图1　任务的状态变化

3.2　uC/OS-II下网络任务的划分

uIP0.9处于网络通信的中间层，其上层协议在这里被称之为应用程序，而下层硬件或固件被称之为网络设备驱动。整个协议的软件结构如图2所示。显然，uIP0.9并不是仅仅针对以太网设计的，并具有媒介无关性。

uIP内核中有两个函数直接需要底层设备驱动程序的支持。一个是uip_input()。当设置驱动程序从网络层收到的一个数据包时要调用这个函数，设备驱动程序必须事先将数据包存入到收发数据缓存uip_buf[]中，包长放到uip_len，然后交由uip_input()处理。当函数返回时，如果uip_len不为0，则表明带有外发数据（如SYN，ACK等）要发送。当需要ARP支持时，还需要考虑更新ARP表示或发出ARP请求和回应。另一个是周期计时函数uip_periodic(conn)，用于驱动网络通信中的所有时钟事件，例如包重发、ARP表的老化[4]。

从本质上来说uip_input()和uip_periodic()在内部是一个函数，即uip_process(u8t flag)，UIP的设计者将uip_process(UIP_DATA)定义成uip_input()，而将uip_process(UIP_TIMER)定义成uip_periodic()，因此从代码实现上来说是完全复用的。

因此将网络通信的实现分为三个任务，分别是网络主任务（NET_MAINTASK）、数据包接收任务（ETH_RE_POLL_TASK）和周期时钟任务（PERIO_TASK）。这三个任务之间相互协调完成数据从链路层到应用层或应用层到链路层的传递。程序中三个主要任务之间的关系和各自的功能如图2所示。

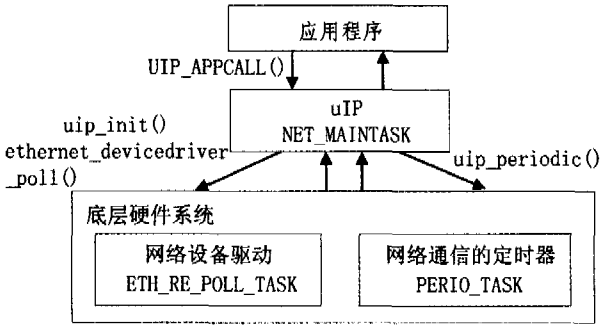

图2　协议软件结构图

数据包接收任务（ETH_RE_POLL_TASK）优先级7，是三个任务中最低的一个任务，它每隔一个时钟节拍20ms执行一次，不断查询是否有新的数据包到，如果有新的数据包到就放到包缓存uip_buf数组里，并置变量uip_len大于零，然后发信号量UipProcess给网络主任务（NET_MAINTASK），并通知它有数据包在uip_buf数组里。ETHERNET_POLL_TASK程序段代码如下所示：

```
void ETHERNET_POLL_TASK(void * pdata)
{
    INT8U err;
    pdata= pdata;
  while(1)
    {
  uip_len=ethernet_devicedriver_poll();//接收以太网数据包
      (设备驱动程序)
if(uip_len>0){//收到数据
  OSSemPost(Sem_UipProcess);//向主任务发送信号}
  }
}
```

网络主任务（NET_MAINTASK）优先级5，在所有通信任务中优先级最高，它完成网络各部分初始化工作后，就负责对新近接受到的数据进行处理并向应用层分发。刚开始它会挂起自己，当接受到数据包收发任务发送的信号后转入运行状态。调用函数uip_process()对收发包缓存uip_buf中的数据进行处理，首先判断数据包是否是IP数据包，如果是IP数据包则查看该包的IP地址，是本子网的IP就更新本地ARP映射表。接下来uip_process()函数会跟据包协议解析出协议类型字段，然后交给相应的icmp_input()、tcp_input()、udp_input()部分处理。uip_process()函数从返回时，如果有要输出的IP数据包，就将其放在数据包缓存里，并置uip_len大于0表示有数据输出。函数uip_arp_out()会为输出的数据包创建以太网包头，网卡驱动函数ethernet_devicedriver_send()最后将数据包发送出去。如果NET_MAINTASK接收到的数据包为ARP包，则调用uip_arp_arpin()函数对ARP请求或者应答包做相应的处理并更新ARP表。NET_MAINTASK程序段如下：

```
void NET_MAINTASK(void * pdata)
{ NET_MAINTASK_init(); //初始化网络模块
  OSTaskSuspend(OS_PRIO_SELF); //挂起任务自身
    INT8U err;
```

```
    pdata = pdata;
    while(1)
      {
    OSSemPend(Sem_UipProcess,0,&err);//收到数据包收发
              任务发出的信号
  if(BUF->type== =HTONS(UIP_ETHTYPE_IP))
  {//是 IP 包吗?
     uip_arp_ipin();
     //去除以太网头结结,更新 ARP 表
       uip_input(); //IP 包处理
       if(uip_len>0){//有带外回应数据
     uip_arp_out(); //加以太网头结构,在主动连接时可能要
       构造 ARP 请求
     ethernet_devicedriver_send(); //发送数据到以太网(设备
       驱动程序)
           }
       }
  else if(BUF->type == = HTONS(UIP_ETHTYPE_ARP))
  {// 是 ARP 请求包
     uip_arp_arpin();//如是是 ARP 回应,更新 ARP 表;如果
              是请求,构造回应数据包
     if(uip_len>0){ //是 ARP 请求,要发送回应
        ethernet_devicedriver_send(); //发 ARP 回应到以太
        网上
        }
  }
```

周期时钟任务(PERIO_TASK)优先级 6,作为网络通信的定时器每隔 100 个时钟节拍运行一次,当周期计时激发,函数 uip_periodic()会查询每一个 TCP 连接的时间状态量,处理连接超时数据包重发事件。PERIO_TASK 的代码段为:

```
void PERIO_TASK(void * pdata)
{
  INT8U I;
  pdata = pdata;
  while(1){
  OSTimeDly(100);//延时 100 个时钟节拍
  for(i=0;i<UIP_CONNS;i++){//UIP_CONN 为 TCP 连接个
                           数
       uip_periodic(i);
       if(uip_len>0){
       uip_arp_out();
       ethernet_devicedriver_send();
    }
  }
}
```

3.3 uC/OS-II 下网络任务的调度

uC/OS-II 是基于占先式的任务调度机制,因此按照任务重要性不同,每个任务被分配了不同的优先级。显然网络主任务是整个网络的得以运转的前提,因此被赋予了比较高的优先级 5。接着当然是网络工作时所需要的时钟任务,同样数据包收发任务被设定优先级 7。至于上层的网络服务则在此基础上进行优先级递增。

然而对于不同的上层网络服务,如 http,ftp 以及 telnet 等应用层协议,却具有相同的重要性。但由于 uC/OS-II 优先级设计的原则,使它们具有了不同的优先级(虽然优先级相近),为了使这些服务得到几乎相同的服务等级,在上层网络服务做完自身的工作后,通过调用 equality_field()函数来决定是否进入延时等待。如果有低优先级网络任务需要运行,则当前任务进入延时等待;否则仍执行当前任务。这样设计使得本为同一优先级的网络任务在人为设定不同优先级的情况下,能协同高效的工作。equality_field()函数的代码如下:

```
INT8U lower_ready;
lower_ready=OSTCBCur->OSTCBY+1;
if((OSRdyTbl[OSTCBCur->OSTCBY]! =OSTCBCur->OS-
TCBBitX)||(OSRdyTbl[lower ready]))
{OSTimeDly(1);
}
```

经过这样的 TCP/IP 协议与操作系统的整合,我们就可以方便地对多个网络任务进行管理,同时也可以方便地向该系统中加入非网络应用的任务。

4 uC/OS-II 下 UDP 协议的实现

uIP 协议栈的重点在 TCP、IP 、ARP 和 ICMP 四种协议,并没有 UDP 协议。而在工控设备的网络通信大多为应答式数据交换,数据流量小,但收发次数多[1]。如果采用面向连接的 TCP 协议,建立连接和撤销连接的开销是非常大的。因此可以采用了 UDP 协议作为基本数据传输方式,而在应用层提供丢失数据的重传的可靠性保障,从而实现数据的高效、可靠的传输机制。文中增加了这一部分的代码,完备了 uIP 协议栈的内容。

UDP(User Datagram Protocol)是建立在网际协议之上的,提供非面向连接得数据传输的传输层协议。UDP 和网际 IP 协议很相似,它们都是采用数据包进行无连接、不可靠的传输。相对于 IP 协议,UDP 唯一增加的功能是提供协议端口,以支持进程间的通信。UDP 数据包处理如图 3 所示。

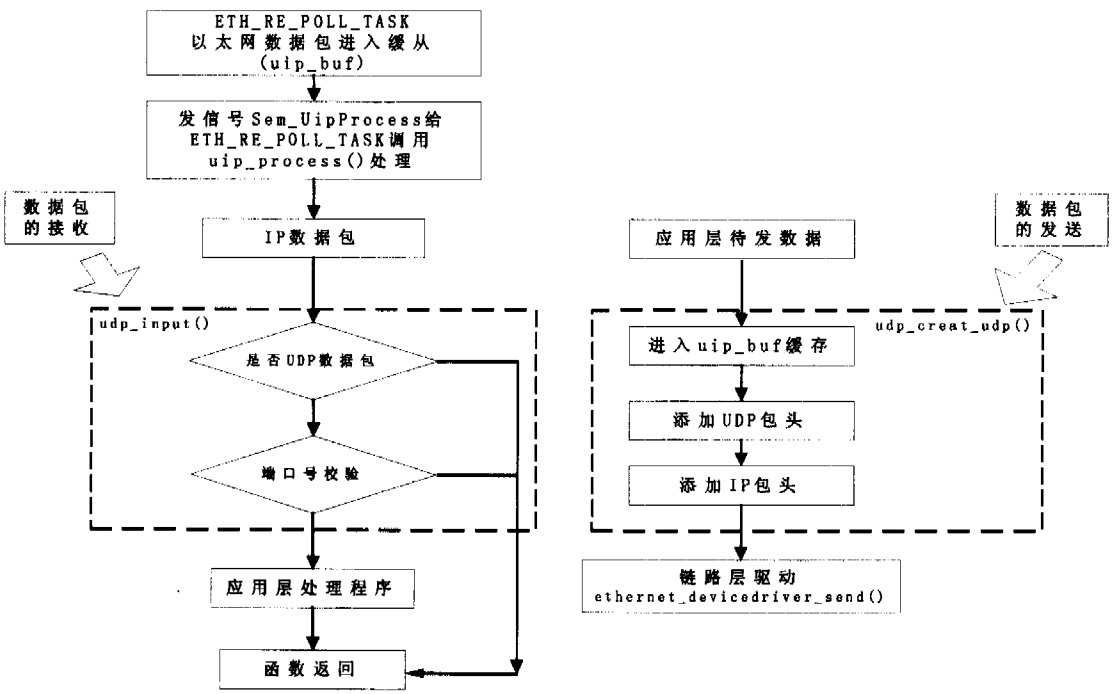

图 3 UDP 数据包的处理流程

(下转第 164 页)

用词例化模板，这类似于 Palmer 实验中采用的规则模板。而表中其他行中的数据是在识别了相应类别下的词类信息后用 CTBL 方法后处理所得的结果。可以看出，词类的作用非常明显，尤其是 NE 和 Factoids 所起的作用。

表 5 词类信息在 CTBL 中的作用

	PK			CTB		
	F-Score	Improved F-Score	Error Reduction	F-Score	Improved F-Score	Error Reduction
CLSP+TBL(Baseline)	.881	——	——	.844	——	——
CLSP+NE+TBL	.921	4%	33.6%	.874	3.0%	19.2%
CLSP+NE+Factoid+TBL	.951	7%	58.8%	882	3.8%	24.4%
CLSP+NE+Factoid+MDW+TBL	.957	7.6%	63.9%	.893	4.9%	31.4%
FMM+ TBL(Baseline)	.898	——	——	.840	——	——
FMM+Factoid +TBL	.929	3.1%	30.4%	.845	0.5%	3.1%
FMM+Factoid+MDW+TBL	.935	3.7%	36.3%	.858	1.8%	11.3%

4.4 分词错误分析

通过认真分析初始分词系统造成的分词错误和加入 CTBL 方法后造成的分词错误，我们大致可以把分词错误分为 4 大类。

①不能准确识别词类的边界。如对于字串李光真图，本来李光真是个人名，而在我们的词类识别时，把李光识别为人名，真图作为一个词典词，从而导致最终的分词错误。

②由于我们所采用的是预先定义好的词类，词类的粒度比较粗，词类的划分也不够准确，有时会产生歧义。如对于实体名词 1999 年内和 1999 年 5 月，词类识别时系统认为它们属于同一词类日期，因此把同样的规则序列作用于这两个词，造成分词错误。

③不能正确识别新词，经常把新词切分为多个单字符。如星巴克被错误切分为星巴克。

④参照语料的不一致性造成分词错误。我们发现在参照语料中有些相同的字串(具有相同含义)在不同的地方却以不同的切分结果出现，这就有可能学得错误的规则。

当然，训练数据比较小也是产生错误的一个不可忽略的原因。

结束语 综上所述，我们提出的 CTBL 方法是非常有效的。当采用 CLSP 系统作为初始分词系统时，经过 CTBL 方法进行后处理后，分词性能在 SIGHAN 举办的第一届中文分词大赛的 4 个标准数据集 PK、CTB、HK 和 AS 上分别排在 2、3、2、1 位。而且该方法还具有一定的鲁棒性。

本文的贡献主要体现在 3 方面：①我们提出一种新的 TBL 方法来解决中文分词问题。通过建立新的词类规则模板，引入有效的语言学信息，我们取得了很好的效果。②我们验证了一些语言学语法信息，如词类、词内结构，对中文分词是非常有用的。③该方法对处理不同分词标准之间的差异是非常有效的。

致谢 在此，我们感谢对本文的工作给予支持和建议的朋友和同学，尤其对微软亚洲研究院自然语言处理组的高剑锋和李沐等人的热情帮助表示衷心的感谢。

参 考 文 献

1 Richard S, Emerson T. The first international Chinese word segmentation bakeoff. SIGHAN 2003
2 Richard S, Shih C. Corpus-based methods in Chinese morphology and phonology. In: COOLING 2002
3 Gao Jianfeng, Li Mu, Huang Chang-Ning. Improved source-channel model for Chinese word segmentation. ACL2003
4 Gao Jianfeng, Wu Andi, Li Mu, et al. Adaptive Chinese word segmentation. ACL2004
5 Wu Zimin, Tseng Gwyneth. Chinese text segmentation for text retrieval achievements and problems. JASIS, 1993, 44(9): 532～542
6 Palmer D. A trainable rule-based algorithm for word segmentation. ACL '97
7 Hockenmaier J, Brew C. Error-driven learning of Chinese word segmentation. In: the 12th Pacific Conference on Language and Information, Singapore, Chinese and Oriental Languages Processing Society, 1998. 218～229
8 Xue Nianwen. Chinese word segmentation as character tagging. Computational Linguistics and Chinese Language Processing, 2003
9 Wu Andi, Jiang Z. Word Segmentation in Sentence Analysis. In: Proceedings of the 1998 International Conference on Chinese Information Processing, Beijing, China, 198. 169～180
10 Wu Andi. Customizable segmentation of morphologically derived words in Chinese. International Journal of Computational Linguistics and Chinese Language Processing, 2003, 8(1): 1～27
11 Wu Andi. Chinese Word Segmentation in MSR-NLP. The first international Chinese word segmentation bakeoff, SIGHAN 2003
12 Brill E. Transformation-based error-driven learning and natural language processing: a case study in Part-of-Speech tagging. Computational Linguistics, 1995, 21(4)

(上接第 74 页)

UDP 数据包的处理程序被写为 API 方式，以在 udp_input()和 udp_creat_udp()实现时调用，其主要的函数为：

```
WORD check_udp(UDPKT * udp, LWORD sip, LWORD dip,
    int ulen);UDP 的校验和
unsigned char udpRead(unsigned char address);UDP 输入
void udpWrite(unsigned char address, unsigned char value);UDP
    输出.
```

结束语 作为一种实时性的嵌入式操作系统，uC/OS-II 内核具有代码公开、可移植、可裁剪的特点，而被广泛地运用到各个领域。但 uC/OS-II 只支持串口通信的局限，限制了 uC/OS-II 支持下的设备的 Internet 的接入。在 uIP 和 uC/OS-II 相结合后，将整个系统移植到一个由 TI 公司的 MSP430F449 和 Cirrus Logic 的 CS8900A 组建的嵌入式 websever 后，系统运行稳定，具有较高的实时性。

参 考 文 献

1 古天龙，健雄. 嵌入式实时系统及其相关问题[J]. 电子商务，1999(12)：12～15
2 Labrosse J J. uc/os-II 源码公开的实时嵌入式操作系统[M]. 邵贝贝译. 北京：中国电力出版社，2003
3 Thomas H. An Introduction to TCP/IP for Embedded Engineers [A]. In: Proc. of the embedded Systems Conf. [C], 2000
4 Dunkels A. uIP-A Free Small TCP/IP Stack[Z]. 2004
5 Postel J. Internet Protocol(IP)[Z]. RFC, 791
6 张懿慧，陈泉林. 源码公开的 TCP/IP 协议栈在远程监测中的应用[J]. 单片机及嵌入式系统应用，2003(8)：18～21