嵌入式协议栈可重构性分析与设计实现*)

宋丽华　张晓彤　王　沁　郭艳飞
(北京科技大学信息工程学院　北京 100083)

摘　要　本文通过对协议栈的可重构性进行分析,设计并实现了一种可重构的嵌入式 TCP/IP 协议栈,并介绍了关键技术的实现细节。该协议栈可以根据用户需求重构为 IPv4、IPv6、双栈协议栈等多种协议栈,具有良好的灵活性、可移植性。该协议栈已成功应用在嵌入式双向通信平台上,通过实际网络测试表现出良好的性能和可靠性。在主频为 66MHz 的 ARM7 CPU 上运行,吞吐率可达 1.5MB/s 以上,目标代码大小仅为传统协议栈的 30%左右,并可满足 IPv4 到 IPv6 网络过渡时期嵌入式设备的网络需求。

关键词　TCP/IP,嵌入式协议栈,IPv6,可重构

Reconfigure Feasibility Analyzing and Implementation of Embedded Protocol Stack

SONG Li-Hua　ZHANG Xiao-Tong　WANG Qin　GUO Yan-Fei
(Information Engineering School, University of Science and Technology Beijing, Beijing 100083)

Abstract　By analyzing the reconfigure feasibility of the protocol stack, this paper design sand implements a reconfigurable embedded TCP/IP protocol stack, and introduces the key technology in detail. This protocol stack can be reconstruct to a IPv4, a IPv6, a dual protocol stack and other kind of protocol stack according to the user request, it has good flexibility and transportability. The protocol stack has been successfully applied to the embedded bidirectional platform, the tests in real network conditions show that the protocol stack provides a good performance and reliability. The throughput is higher than 1.5MB/s when run on ARM7 CPU which clock frequency is 66MHz, the object code size is only 30% of traditional ones or so, and can meet the request of the embedded system for connect to the internet during the transitional period from IPv4 to IPv6.

Keywords　TCP/IP, Embedded protocol stack, IPv6, Reconfigurable

1　引言

随着嵌入式系统和网络技术的迅猛发展,以 Internet 为介质实现信息交互的要求越来越强烈。据专家预测,今后 70%的 Internet 信息传输来自于嵌入式设备。然而,嵌入式系统计算资源、存贮资源等十分有限,采用传统通用的协议栈无疑无法满足这些方面的要求;一些专用嵌入式操作系统,如 VxWorks、uCLinux、pSOS、VRTX 等操作系统也提供了自己的专用嵌入式协议栈,但都是以其具体开发的实时操作系统为平台,没有考虑不同软硬件资源,移植困难;另外,IPv4 地址资源受到严重挑战,IPv6 网络代替 IPv4 网络已成为必然。目前一些开源嵌入式协议栈如 uC/IP, lwip, uip, TinyTcp 虽然具有代码量比较小、占用系统资源较少,具有较好可移植性,但都不支持或者没有完成 IPv6 协议栈的开发,不能满足过渡时期嵌入式设备的需求[1]。

如何开发代码小巧、资源占用量低、可移植性好、支持未来协议扩展的灵活而高性能的嵌入式 TCP/IP 协议栈,成为了极具挑战的研究课题。本文针对嵌入式 TCP/IP 协议栈开发的关键问题,提出了一种可重构嵌入式协议栈实现方案。该协议栈不仅能够根据实际嵌入式设备网络接入环境配置成 IPv4、IPv6 或 IPv4 和 IPv6 双栈 TCP/IP 协议栈等多种协议栈,使设备在拥有网络通信能力的兼备灵活性和支持下一代网络应用的扩展性,并大大缩减了协议栈代码量和内存空间占用。同时,该协议栈提供了通用网络接口,使其能够支持不同软硬件结构,具有平台无关性和良好的可移植性。

2　嵌入式协议栈可重构性分析

嵌入式可重构协议栈设计的目的是:通过对协议栈可重构性进行分析,精简实现主要协议,找到合适的拆分单元,在减少系统资源占有量、增强灵活性的同时尽量提高系统的处理速度,保证高吞吐率。

2.1　协议组合可重构性分析

首先,不同层间协议相互独立。由于 TCP/IP 协议在制定的时候就按照一种层次的思想,不同层次的协议需通过简单的层间接口进行服务交互。并且,当任何一层发生变化时,只要接口关系保持不便,则该层上、下各层均不受影响。其次,同层间协议或者相互独立,或者和同层间其他协议协同工作。传输层的 TCP 和 UDP 协议分别为面向连接服务和无连接服务两者之间没有任何交集,互不影响,各自独立工作。IP 层的 IPv4 与 IPv6 及相关配合使用的协议可以在一个协议栈中并存,因为 IPv6 采用了新的报头格式和 128 位地址,处理上各自独立。这样,使得不同层间协议的耦合度很低,便于协议的删减和扩展,使协议栈重构成为可能。采用模块化思想来实现,使用户可以需求选择协议,构造特定需求的协议栈。

*)北京市科技重大项目"交互式数字电视信道传输核心技术开发"(京科技发[2002]188 号)、北京市科技产业化项目"SOC 设计服务及重点产品关键技术研究"(课题编号 D0306008041021)。**宋丽华**　博士研究生,主要研究方向为嵌入式系统与通信等。

不考虑应用层协议，我们用集合表示不同层次的协议进行：

$S_1=\{x|x \text{ is a transport layer protocol}\}$，

$S_2=\{y|y \text{ is a network layer protocol}\}$

定义规则：1)各层之间是独立的。2)高层协议若被选通，则需有底层相应协议的支持。3)若依赖其它协议协工作的协议被选通，则相关协议必须被支持。可得出用户可重构出的协议栈类型总数 N 为：

$$N=\sum_{i=0}^{m}\sum_{j=1}^{n}C_m^iC_n^j \quad (m \text{ 为传输层可划分协议组个数，} n \text{ 为 IP 层可划分协议组个数}) \tag{1}$$

例如：设 $S_1=\{P_1,P_2\}$，$S_2=\{P_3,P_4,P_5\}$，其中 $P_1\cap P_2=\Phi$，$P_3\cap P_4\cap P_5=\Phi$，且 $P_4\cap P_5\neq\Phi$(注：交集为空，表示两个协议各自独立工作，不为空表示协同工作)，则按照以上三条规则，可构成的协议栈为：$\{P_1,P_2,P_3,P_4,P_5\}$，$\{P_1,P_2,P_3\}$，$\{P_1,P_2,P_4,P_5\}$，$\{P_1,P_3,P_4,P_5\}$，$\{P_2,P_3,P_4,P_5\}$，$\{P_1,P_3\}$，$\{P_1,P_4,P_5\}$，$\{P_2,P_3\}$，$\{P_2,P_4,P_5\}$，$\{P_3\}$，$\{P_4,P_5\}$。

2.2 资源有效可复用性分析

在协议栈所占用的空间里，由于大量数据的流动和处理，缓冲区占用相当大的份额。如果每层都开辟单独的存储区来存放数据，不仅会造成资源的浪费，并且大量的数据拷贝会大大降低处理速度[2]。考虑到网络数据的特点，每层处理数据时只是处理相关的首部，高层的数据原封不动地传给高层。见图 1 所示，可以设计一种缓存管理机制，采用复用缓存区方案解决这一问题。设采用这种管理方式占用系统资源为 S_{code}，节省的空间为 S_{save}，造成的系统处理时间为 T_{code}，节省的系统处理时间为 T_{save}，我们下面对其可行性进行分析。

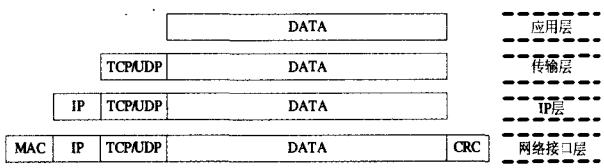

图 1　网络数据格式示意图

MAC 帧首部长度为 14 字节，CRC 校验 4 个字节，IP 报文头通常情况下为 20 字节，TCP 首部 20 字节(UDP 首部 8 字节)，按默认链路 MTU 值为 1518 字节，则 DATA 长度不会超过 1460 字节。如果用 l_{share} 和 $l_{separate}$ 分别表示共享缓存空间占用的总字节数和各层独立缓存所占用的总字节数，那么采用共享缓存节省的内存空间百分比 p 为

$$p=1-\frac{l_{share}}{l_{separate}}*100\% \tag{2}$$

假设一包长度为 1518 字节的数据包，由(2)式可知，最大可节省约 75%的缓存空间，即 4.5k 的 RAM 资源。当有多包数据在协议栈中处理时，可节约更多的 RAM 资源，而提供这样的管理机制代码量不过百余行，目标代码也就 1k 左右，显然 $S_{save}>S_{code}$。

采用共享缓存区便于实现层间数据零拷贝，即当数据在层间传递时，可以通过指针移动来避免数据的拷贝。理想状态下，只考虑指令的基本执行时间。假设：1)移动一次指针操作需要耗费 2 个时钟周期；2)一次数据拷贝耗费 2 个时钟周期；3)总线长度为 32 位；4)数据长度为 n 字节，则一次可以拷贝 4 字节数据，那么零拷贝与一次拷贝所耗费时间之比为

$$C(\frac{z}{s})=\frac{2}{2n/4}=\frac{4}{n}(0<n<1518) \tag{3}$$

通常情况下，$n>34$(链路层首部长度 14+IP 层首部长度 20)。由式(3)可知，当数据长度较大并且经过多层拷贝时，$T_{save}\gg T_{code}$。由以上分析可知，对缓存区进行复用，可以降低 RAM 资源占用率并且降低系统时间开销。

3 可重构嵌入式协议栈体系结构设计与实现

3.1 系统总体体系结构设计

根据以上分析和实际需求，将设计目标定为：1)具有比较小的代码量，尽量少地占用系统资源，高效、稳定、易于移植；2)支持 IPv6 协议，使嵌入式设备具有更好的可扩展性；3)具有可重构性，嵌入式系统可根据自己的实际需要在编译的时候对协议栈进行重构，解决通用协议栈不能适应嵌入式系统专用需求的矛盾。

采用模块化的设计思想、C 程序设计语言。并且，为了便于移植和减少系统的开销，我们不考虑任何嵌入式操作系统基于裸机实现。系统体系结构设计如下：其中各个模块均由宏开关打开，尽量减少各模块间的耦合，使协议栈可重构为 IPv4、IPv6 或双栈 TCP/IP 协议栈，也可以根据需求重构为 UDP/IPV4、UDP/IPV6 等协议栈。系统的体系结构设计如图 2 所示。

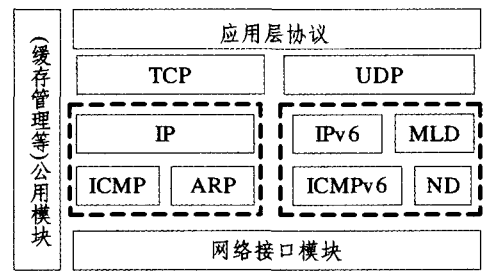

图 2　可重构 TCP/IP 协议栈体系结构示意图

公用模块主要包括：缓冲区管理模块、校验和计算以及字节顺序转换函数。为协议栈系统提供高效稳定的缓冲区管理机制，并为一些通用且处理频繁的操作提供调用函数，以使协议栈达到高效、稳定、简洁的目标。

网络接口模块是连接协议层和物理硬件的中间层，为网络协议提供统一的发送接口、屏蔽不同的物理介质，同时负责接收并处理链路层帧。网络接口的这种屏蔽作用可以使协议栈实现时不必过多考虑底层环境，在不同系统间移植时可以减少改动甚至不用改动。

IP 层对接收的数据包通过判断以太网类型字段是 0x8600 还是 0x86DD 将数据传送到 IPv4 或 IPv6 模块进行处理，可以通过宏开关构建 IPv4、IPv6 或双栈协议栈，其中 ICMP，ARP 为与 IPv4 对应的相关协议。在 IPv6 中，ICMPv6 具有 IPv4 的 ICMP 常用功能，并且用邻居(ND)发现取代了 IPv4 的地址转换协议 ARP、用多播侦听发现(MLD)取代了 IPv4 中的网际控制协议 IGMP。IP 层主要功能是寻址，还有分段和重组。IP 还在必要时维护接口的多播组。因为在 IPv6 中，广播已经被多播取代，而且 ICMPv6 的邻居发现等都依赖于多播[5]，故 IPv6，ICMPv6，MLD，ND 为一组相关协议。

UDP 协议主要是对报文进行校验以及应用层协议寻址，并将高层的数据进行封装，然后将数据传送到 IP 层进行处理，比较简单。TCP 协议比较复杂，包括协议状态机的处理、RTT 估算、快速恢复和快速转发等主要功能，TCP 和 UDP

两个模块也可以通过宏开关来根据需要进行选择编译。考虑到效率和代码量的关系没有提供类似 BSD Socket 的用户接口函数，通过设计协议控制块(PCB)数据结构解决与应用层协议的接口、管理和控制问题。

3.2 关键模块的实现

3.2.1 缓存管理模块

高效的缓冲区管理机制需要满足以下几个条件：1)能够迅速为到来数据分配空间。2)方便变长缓存操作。3) 尽量减少层间数据拷贝。本文采取一种预分配机制，即先静态地从内存中划分出一块大小固定的内存空间，然后在此固定内存空间上进行内存的再分配。这样，防止协议栈耗尽所有系统内存，提高系统的可控性和可靠性。采用变长和定长两种分配方式，一部分用作固定大小的静态缓冲块组，另一部分用作动态缓冲区。其中静态缓冲部分采用固定大小的内存块，位图管理方式，用来接收数据。这种分配方式在编译或者链接时已经将指定的内存空间分配好，不会出现分配失败的情况，且速度快，避免接收数据时产生瓶颈，保证了嵌入式系统对实时性的要求。动态部分采用最先适应分配算法和双向链表进行管理，主要用于发送数据时分配内存空间，方便变长缓存和数据移动操作，减少数据的拷贝操作，具有可动态分配、回收等特性，大大提高系统的内存利用率(参考 BSD4.4BSD mbuf 设计[1])。管理模块的数据结构如图 3 所示。

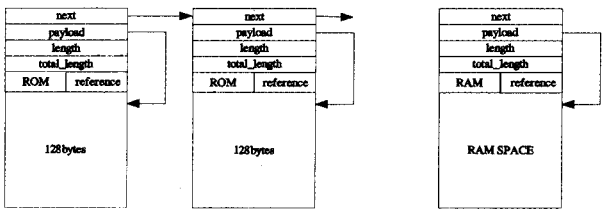

图 3 MYBUF 缓冲池

next 字段指向下一个 MYBUF；payload，指向 MYBUF 数据区；total_length 等于当前 MYBUF 及其后每个 MYBUF 的 length 字段的总和。当到来的数据被分配到几个缓冲块中时，该字段可以保证包的完整性。flags 表示 MYBUF 的类型是静态缓冲区 ROM 还是动态缓冲区 RAM。如果是动态缓冲区，length 和 total_lenth 的值是相等的。reference 是一个引用计数器，它的值标志该 MYBUF 被引用的次数。当有新的引用(指针)指向该 MYBUF 时，对应的计数器增加；当原有的引用取消时计数器减少，当对应 MYBUF 的计数器值变为零时才可以释放该 MYBUF 所指向的缓冲空间。对于 ROM 类型的 MYBUF，当 reference 字段为 0 时，表示该静态缓冲块可用，可以通过该字段完成静态缓冲块的分配和回收。

通过 next，payload，length，total_length 等字段的控制可以保证：当数据在协议栈中处理时，只通过指针移动操作传递数据，并不做实际的数据拷贝，各层间传递的都是数据指针。只有当数据最终被驱动程序发送出去或是被应用程序取走时，才进行真正的数据搬移，避免了数据的多次拷贝。静态缓冲区单位块大小、动态缓冲区大小都是可配置的，任何系统可以根据网络和系统资源的实际情况对其长度进行设置，提高了系统的资源利用率和该内存管理机制的通用性。

3.2.2 网络接口模块

IPv6 在地址上相对于 IPv4 做了很大优化和改动，这些变化也对协议栈和网络接口的实现方法产生了一定影响。例如：IPv6 要求一个物理接口支持多个逻辑接口。这对于 IPv4 中可能不是必需的，但是 IPv6 网络中一个物理接口的 IP 地址可能对应链路本地地址、本地站点地址和全球单播地址三类地址。另一方面，IPv6 和邻居发现 ND (Neighbor Discovery) 协议，要求对网络接口进行修改，以实现新的组播地址映射和地址解析等功能。考虑到以上情况，将网络接口抽象为一个 netif 结构体，对网络接口层进行统一的控制和管理以及完成数据的发送和接收。具体数据结构如图 4 所示。

next 指向链表中的下一个网络接口，即支持一个物理接口对应多个逻辑接口，每个网络接口对应一个 netif 结构，每个 netif 结构记录该网络接口配置、统计信息和全局参数；ip_address 指向本接口的 IP 地址链的首指针；mac_addr 为一个数组存放接口的 MAC 地址；multi_addr 指向 MAC 多播地址链首地址；input 是一个函数指针，采用“软中断”的方式接收数据，即高层协议在网络接口层注册接收函数的函数指针，由底层驱动程序调，这样可以将数据传到高层。Output 和 input 一样，也是一个函数指针，由 IP 层发出报文时调用。该函数先查找与目的 IP 地址对应的 MAC 地址，然后发出报文；linkoutput 也是一个函数指针，与 output 函数不同，该函数在链路层使用，直接发出报文，即链路层首部已填充好，所以无需再寻找地址对应关系；name 用于标志接口类型；num 用于区分多个相同类型的接口，如有两个以太网接口时，name[2] 的值都为 et，num 则分别为 0、1，则这两个接口可标识为：et0、et1；省略了一些统计信息变量。网络接口屏蔽了硬件细节，在不同系统间移植时也可以少改动甚至不用改动。

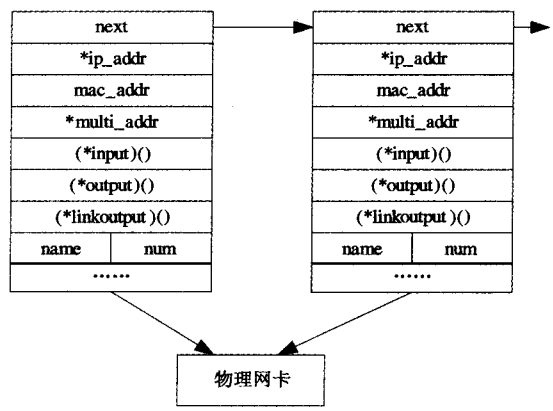

图 4 网络接口

3.2.3 协议控制块

考虑到效率和资源有限的关系没有提供类似 BSDSocket 的用户接口函数，通过设计协议控制块(PCB)数据结构解决传输层状态控制、参数统计，以及和应用层协议的接口、管理和控制问题。以 UDP 层的 PCB 实现为例，见图 5，udp_pcb 采用链表管理，内存空间由协议栈缓存管理控制和分配。并且系统可根据实际需求对控制块的个数进行配置，避免空间的浪费。

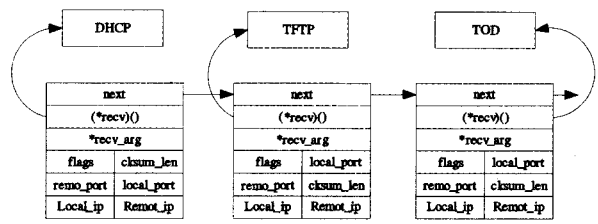

图 5 PCB 链表及工作原理

我们可以看到 udp_pcb 记录了报文的所有基本信息，next 指向下一个 udp_pcb；一个 UDP 会话由 IP 地址和端口号来标志，这些信息分别存储在 local_ip，dest_ip，local_port，remote_port 字段中；flags 字段表示 UDP 在该会话中的校验策略，如果该值为零则不进行校验；chksum_len 为校验的数据长度，当有伪首部的时候该字段的使用就变得有实际意义；recv 函数指针和 netif 中的 input 一样用于层间协作，我们称之为回调函数，它的功能是帮助 udp 协议将数据传递到对应的应用程序中。应用层的协议要有一个 xxx_recv()函数用来处理接收的数据，并且函数的参数、返回值类型要和 pcb 中回调函数参数保持一致，在初始化时调用 udp 层的绑定回调函数的 udp_recv()将函数注册，udp 层建立相应的 pcb 块到其所维护的链表中，这样当有该协议的数据到来时，udp 层查询所维护的 pcb 链表，核对其中的 local_port 和 remote_port，如果匹配则可以通过其注册的处理函数的函数指针，帮助 udp 协议将数据传递到对应的应用程序中。

通过回调函数可以使用函数指针以使下层协议能够调用上层协议的功能，但是仍保持协议分层结构的完整不至于相互影响。这样，可以不通过 socket 调用，使用户根据需要编写应用协议，通过实现约定好的简单函数，扩展协议栈的功能。

4 测试及性能分析

4.1 功能测试

4.1.1 Window 测试平台

在 X86CPU，WindowsXP 操作系统，Visual C++编译器下搭建仿真测试平台，使用第三方软件 WinPcap 来为协议栈从网卡接收和发送数据报文，它能为 Win32 应用程序提供独立于主机协议而发送和接收原始数据报文的能力。协议栈设置：RAM 空间设为 10k，20 个 128 字节的静态缓冲块，测试结果如表 1 所示。

表 1 Window 平台下功能测试结果

类型	测试环境	结果
TCP/IPv4 协议栈	编写高层客户端应用程序 DHCP、TFTP、TOD 等协议，通过双绞线与局域网相连，并在其它 PC 上安装相应应用协议的 server 端。	执行 DHCP、TFTP、TOD 等客户端程序可为本协议栈获取 IP 地址，并对 IP 地址进行维护；put，get 文件操作正确，且可以从局域网的 time server 处获取正确的 UTC 值。
TCP/IPv6 协议栈	在同一局域网中与已安装 IPv6 协议的 Win XP 系统之间通过对等线相连，形成一个单独子网，在其中一台 PC 的 Windows 操作系统下搭建的嵌入式协议栈测试平台，编写测试程序进行测试。	使用网络管理和监视工具 commview 可以连续捕获到 IPv6/ICMPv6 request、reply 报文以及邻居发现协议的邻节点请求、公告报文。
IPv4/IPv6 双协议栈	通过编写测试程序，对双协议栈正确性进行测试。	协议栈可以根据到来的数据进行匹配将不同类型的 IP 协议分配到不同的协议栈进行处理，执行 ping6、DHCP、TFTP 等程序操作正确。

4.1.2 嵌入式测试平台

嵌入式系统测试平台见图 6。由 ARM7CPU、双向信道的物理层调制解调芯片 Jupiter 及 MAC 协议处理器 FPGA 组成的 HDTV 双向系统上，根据 DOCSIS1.1 射频接口规范[3]，CM 内置协议栈传输层只需要支持 UDP 协议即可。因此，将协议栈配置为嵌入式 UDP/IPv4 协议栈，移植到该测试平台上运行。RAM 空间设置为 5k，16 个 128 字节的静态缓冲块。

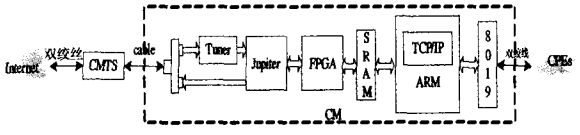

图 6 嵌入式双向系统测试平台

结果：CM 可以快速正确地完成 IP 地址获取、TFTP 下载配置文件、TOD 日期时间获取以及 IP 连接维护等操作，并且可连续工作数十个小时，稳定可靠。

4.2 性能测试

4.2.1 吞吐量测试

借助与嵌入式系统平台相连的 CPE 上网络工具对协议栈的性能进行测试、运行网络的 ping 程序测试。不同长度的数据包分别执行 100 次，取平均应答时间，测试结果和参数如表 2 所示。由于试验环境限制，可以看出嵌入式协议栈的处理吞吐量可大于 1.5MB/s，可以满足大多数嵌入式系统的需求。

表 2 ping 测试统计数据

Ping 数据包长度 (bytes)	平均应答时间 (ms)	丢包率 (%)	吞吐量 (Kbytes/s)
128	0.232	0	551.7
256	0.286	0	895.1
512	0.418	0	1224.9
1024	0.679	0	1508.1
其他参数	处理器类型：ARM7TDMI 内核时钟频率：66MHz		

4.2.2 目标代码大小比较

表 3 目标代码大小比较

	UDP/IPv4 协议栈	TCP/IPv4 协议栈	TCP/IPv6 协议栈	IPv4/IPv6 双协议栈	Linux2.0.37 TCP/IP stack
目标代码大小 (kbytes)	19	35	36	41	117
与 Linux2.0.37 TCP/IP stack 相比	16.2%	30.0%	30.8%	35.0%	100%

注：TCP/IPv4 协议栈中包括 UDP 协议，TCP/IPv6 协议栈亦然，但 UDP/IPv4 协议栈中不包括 TCP 协议。

将协议栈配置为不同类型协议栈，在 X86 下 gcc 编译。未标识版本信息的协议栈为本嵌入式协议栈。和 Linux2.0.37TCP/IP 协议栈代码目标文件大小[4]进行比较，由表 3 可

（下转第 76 页）

协议、移动采用 CMPP 协议。网络管理监控中心通过以上协议把收集到的网络告警、流量信息通过以上协议传送到短信中心，短信中心再通过无线渠道下发这些告警监控信息到终端。由于短信中心到终端侧目前都已很成熟并形成规范，这里不再详述，本文只是介绍网管中心和短信网关之间的接口，以电信标准协议 SMGP 来阐述其流程。

电信网关通信协议 SMGP 是一个基于数据包的交互式协议，通过 TCP/IP 传递数据。每个数据包都包含请求标识，代表数据包的用途。服务提供商与短信网关或者短信网关与短信网关之间采用客户-服务器的方式交互信息，客户向服务器发送一个请求包，服务器向客户返回回应包。客户发送的所有请求包都包含一个唯一的序列号，服务器返回的回应包也带有相应的序列号，以便客户识别这是哪一个请求的回应。客户和服务器之间采用长连接，如果在一定时间内客户和服务器之间没有发送合法的请求包，连接就中断网管服务器首先以发送者或者是收发者身份登录 SMGP 后，就可以发送短消息。消息流程如图 5 所示。

实现流程：

(1)当网管中心监控到设备的告警信息后实时调用短信模块发起短信呼叫，短信的内容为监控到的设备告警信息；

(2)网管中心通过 SMGP 协议的 SUBMIT 消息向短信网关发送该告警消息；

(3)短信网关收到短信后判断计费用户的计费类型，如果计费用户是后付费用户，则转流程(5)；

(4)如果计费用户为预付费用户，则短信网关通过 SMGP 协议扩展计费接口向预付费平台递交查询用户状态请求 Query_UserState；

(5)预付费平台返回 Query_UserState_Resp；如果 Query_UserState_Resp 表明用户状态正常并且余额足够支付本次下行短信费用，则转流程(5)，如果 Query_UserState_Resp 表明计费用户为非预付费用户，则按后付费方式对该用户进行计费处理；如果 Query_UserState_Resp 表明用户状态不正常(不包括非预付费的情况)，则二级短信综合网关向网管中心返回失败应答，并产生失败记录，网管中心到终端的短信发送流程结束；

(6)短信网关通过 SMGP 协议的 SUBMIT_RESP 向网管中心返回正确响应；

(7)短信网关根据被叫终端号码对短信进行路由，通过 SMPP 协议的 SUBMIT_SM 指令向 SMC 短信中心发送短信；

(8)SMC 短信中心返回短消息应答 SUBMIT_SM_RESP；

(9)SMC 短信中心向终端用户发送该包含告警消息内容的短信；

(10)用户终端向 SMC 短信中心发送短消息成功/失败接收的响应；

(11)用户终端接收短信成功/失败后，SMC 短信中心产生回执，并通过 SMPP 协议的 DELIVER_SM 消息将该消息发送给短信网关；

(12)短信网关接收到回执后向 SMC 短信中心返回 DELIVER_SM_RESP 消息；

(13)对于预付费用户，短信网关收到 SMC 短信中心返回的成功回执后，通过 SMGP 协议扩展计费接口向预付费平台递交扣费请求 Payment_Request；

(14)预付费平台将 Payment_Request_Resp 返回给短信网关，短信网关根据扣费成功结果产生成功话单，根据扣费失败结果产生失败回执；

(15)对于后付费用户，短信网关收到 SMC 短信中心返回的成功回执后，产生成功话单，收到 SMC 短信中心返回的失败回执后，产生失败记录。

鉴于维护角度考虑，建议网管中心和短信网关之间不需要状态报告和计费流程包。

SMGP 协议的具体 submit 消息包请参考中国电信 SMGP3.0 标准协议。

结束语 本文提出的利用移动终端实现网络管理与当前的网络管理技术方法相比更能体现下列优势：

1) 管理范围更广。网管人员可以在任何地方、任何时间，通过移动终端就可以监测和控制被管理的网络，它不仅仅限于网络工作站。

2) 移动性更好。本系统把移动终端作为工作站，体现了更好的移动性。

3) 可移植性好。用户可以在原有网络管理平台上进行升级，无须投入过多的成本，就可以实现真正意义上的网络分布式管理；

参 考 文 献

1 雷雪梅主编. 苏力萍，等编著. 现代网络管理. 北京：国防工业出版社，2005
2 万加富，张文斐，张占松编著. 网络监控系统原理与应用. 北京：机械工业出版社，2003
3 郭金发，张龙编著. 短信与 BREW 开发技术及实践. 西安：西安电子科技大学出版社，2005
4 曹建编著. WAP 编程与开发实例教程. 电子工业出版社，2001
5 宋俊德，王劲松. 无线移动终端现状与未来. http://it.cpst.net.cn/mob/2006-03/1143708067.html
6 吴晨光，高艳娟，朱小兵. 基于 WAP 技术的智能住宅远程监控系统实现. 计算机应用与软件，2004，21(7)
7 李捷，王汝传. 基于 WEB 平台的分布式网络管理模型的研究与实现. 计算机工程与应用，2003，36：137
8 WAP PUSH 业务入门. 诺基亚论坛，2001
9 范绍山. WAP 中推送技术的分析与设计. 论文联盟. http://www.lwlm.com/show.aspx?id=27019&cid=25

(上接第 72 页)

知，本文设计的嵌入式协议栈的目标代码大小仅为传统协议栈的 16%～35%，并且在不需要支持 TCP 协议时，系统可节约 46%左右的协议栈内存空间，可以大大节省系统资源。

结论 测试结果表明，本文设计的可重构嵌入式协议栈，可根据实际网络环境配置成 IPv4、IPv6 或双栈 TCP/IP 协议栈等多种协议栈，具有较高的吞吐量和较小的目标代码量，并且易于移植。通过实际网络测试表现出良好的性能和可靠性，可以满足大多数嵌入式系统的网络需求。

参 考 文 献

1 Wright G R, Stevens W R. TCP/IP Illustrated. The Implementation Vol 2. [M]. MA: Addison-Wesley, 1995
2 刘飞，芦东昕，缪敬. 面向通信领域通用内存管理单元的算法和实现[J]. 计算机工程，2003，29(22)：80～82
3 Cable Television Laboratories Inc. Data-Over-Cable Service Interface Specifications—Radio Frequency Interface Specification[S], 2002
4 Li Yun-Chen, Chiang Mei-Ling. LyraNET: A Zero-Copy TCP/IP Protocol Stack for Embedded Operating Systems. In: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications(RTCSA'05)[C], 2005
5 Davies J. Understanding IPv6[M]. Microsoft Press, 2002