摘 要

Internet 以及嵌入式系统是目前最为常用的，将它们应用于现代化智能测试领域，使测控系统的现场技术与计算机网络的主流技术很好地融合起来，不但可以打破国外厂商的垄断企图，而且还可共享规模经济的效益，大大降低产品开发和系统组建的成本。随着计算机、通信及网络技术的发展，实现大量智能电子设备与 Internet 的结合，即电子设备 Web 化是信息时代的新趋势。本论文以数据结构和信息传递为线索，利用 SOC 单片机扩展网络接口芯片，在硬件和软件上进行了系统设计，实现了瘦服务器的功能。文中还给出了瘦服务器的应用实例。

本论文首先给出了网络服务器和瘦服务器的概念，接着在比较了现有的几种嵌入式 Internet 技术的基础上，提出了单片机系统作为瘦服务器实现大型印刷机控制的新方案。接着详细介绍了本方案中瘦服务器的硬件构成、以太网接口芯片的驱动方式，着重分析和阐述了嵌入式 TCP/IP 协议栈的构成、每个协议的内容、实现这些协议的各个程序模块以及其中关键的编程细节等，并配有相关的程序流程图加以详细说明。同时，论文也介绍了图形化编程语言 LabVIEW 设计的客户端软件，它使网络化虚拟仪器和远程控制成为可能。

本课题的研究工作以建立单客户机多服务器的先进工业测控网络为目的，重点就网络中的瘦服务器进行了研究和设计，具有一定的先进性和广阔的应用前景。

关键词：嵌入式 Internet，瘦服务器，TCP/IP 协议，单片机，虚拟仪器

Abstract

Internet as well as the embedded system is common now. The application of these two techniques in modern measurement&control field makes the techniques of measurement control system and the network fused well. It can not only break the attempt of monopoly from abroad manufactories, but also decrease the cost of research new products and the construction of the system. From the paper, it can be concluded that it is necessary to make many intellectual electronic equipments access the Internet in the new era with the development of the computer, communication and the network technique. With the thread of the data structure and the communication of the information, this paper designs the hardware and software of the system and implements the functions of the thin server by using the SOC SCM extended network interface chips. Moreover, the paper gives an application example of the thin server.

This paper gives the conception of the network server and the thin server first Based on the comparison of several current embedded Internet technologies, this paper gives the new schemes to control the large-scale printer by using the SCM system as the thin server. In the following part of this paper illustrates the thought and method of thin server with hardware and software. It illustrates the construction &function of hardware, the driving method of Ethernet interface chip, and analyzes the construction of embedded TCP/IP protocol, the content of each protocol, and each function module and programming details with flow charts.At the same time, a kind of graphical programming language LabVIEW is introduced. It makes network virtual devices and remote control possible.

The purpose of the research work is to upbuild modern industry measurement& control network whose structure is single client and many servers The thin server is rescarched and designed in this work, which has better performance and wide apply foreground.

KeyWords: Embedded Internet, Thin server, TCP/IP protocol, SCM, Virtual Instrument

第1章 绪论

1.1 论文选题的背景及意义

1.1.1 嵌入式 Internet 技术的发展及趋势

所谓嵌入式 Internet 是指电子设备通过嵌入式模块而非 PC 系统直接介入 Internet，以 Internet 为介质实现信息交互的过程。Internet 技术的飞速发展带来了一个新时代。在讨论嵌入式 Internet 技术之前，回顾一下 Internet 的发展和应用历史是有意义且很必要的。

Internet 的发展与应用经历了并且正在经历着如下几个阶段：

（1）Internet 发展的萌芽阶段（1980-1990 年）。

在这一阶段，实现了异种计算机之间的联网和信息交换，采用 TCP/IP 协议进行各种信息的交换，主要解决专业领域如银行、军用系统采用不同操作系统的大、中、小型计算机的联网问题，这时并不称之为 Internet，TCP/IP 网络仅是少数计算机专家的概念，这是 Internet 发展的萌芽阶段。

（2）PC 机作为客户机，Internet 席卷全球（1990-2000 年）。

在 TCP/IP 协议网络之上发明的 email 和 www 普遍应用，Internet 国际互联网被大众接受，这时一个重要的条件是 PC 普及得非常广泛，即形成了 Client/Server 体系结构（客户机/服务器），进而发展为 Browser/Server 结构（浏览器/服务器），这时的客户机是已广泛普及的 PC，而服务器是相对复杂的，价格昂贵的超级计算机，即我们所谓的“胖服务器”。在这一时期，Internet 象一场革命改变了世界。

（3）嵌入式 Internet 时代（2000-2010 年）。

在这一阶段，人们将给地球披上一层“电子皮肤”，嵌入式片上系统成为瘦服务器。预测未来 Internet 将向何处去，这是全世界科学家关心的问题，包括美国贝尔实验室总裁 Arun Netravali 的一批科学家对此做出了预测：在这阶段“将会产生比 PC 时代多成百上千倍的瘦服务器和超级嵌入式瘦服务器。这些瘦服务器与我们这个世界你能想到的各种物理信息、生物信息相联接，通过 Internet 自动地、实时地、方便地、简单地提供给需要这些信息的对象”。

1.1.2 嵌入式 Internet 技术的应用前景

嵌入式系统接入 Internet 技术具有广阔的应用前景，其主要应用领域可以包括：

1．工业自动化。工业现场应用了大量的 8、16、32 位嵌入式微控制器，其网络化是提高生产效率和产品质量、减少人力资源的主要途径，在工业过程控制、电力系统、电网安全、电网设备监测、石油化工等系统中具有广阔的应用前景；

2．智能大厦。智能大厦是信息（自动化）技术与房地产相结合的产物。随着计算机的普及应用以及网络和自动化技术的发展，大厦内所有公共设备都将采用嵌入式智能系统来提高大厦的服务能力，嵌入式系统的应用可实现大厦内各种操作和信息的共享，实现按需控制；

3．智能仪器。参考 PC 机主板设计方法设计的嵌入式计算机智能仪器应用系统通用性强，并可根据不同要求，选择其中的全部或部分电路，只需对软件做部分修改或删除，简化系统，大大提高了开发效率，降低了开发成本，具有广阔的应用空间；

4．信息家电。21 世纪是数字化网络的时代，计算机、通信、控制、3C 技术相互融合，各类消费类电子产品随着数字化技术与互联网的普及而走向数字化、网络化。嵌入式计算机是实现电子产品数字化、网络化的技术基础。随着数字化进程的日益加深，人们的家居生活越来越明显地印上了“网络的烙印”，最突出地表现就是家电开始走向数字化、网络化。

嵌入式测控系统与 Internet 结合具有以下优点：

（1）速度越来越快，可以满足各种数据传送的实时性要求；

（2）Internet 传输的信息更加丰富，不仅可以传输数据信号，还可以传输声音和图像，信息是以数据、文字、图像、表格或语音等形式反映在网页上，数据的更新速度更快，可操作性更强；

（3）协议是公开的，网页上的信息可以更加方便地读取；

（4）实现了完全的分散控制，系统的功能更强大，可靠性更高，具有更加优良的性价比。

1.1.3 服务器以及瘦服务器

在网络技术和应用快速发展的今天，作为网络核心的服务器其重要性日益突出，网络时代为服务器的应用提供了广阔的空间，服务器因此进入了技术、应用和市场互动并迅速发展的新阶段。

服务器是 20 世纪 90 年代的迅速发展的主流计算机产品，它是在网络环境下提供网上客户机共享资源（包括查询、存储、计算等）的设备，具有高可靠性、高性能、高吞吐能力、大内存容量等特点，并且具备强大的网络功能和友好的人机界面。服务器首先是计算机，只不过是能提供各种共享服务（网络、Web 应用、数据库、文件、打印等）的高性能计算机，它的高性能主要体现在高速度的运算能力、长时间的可靠运行、强大的外部数据吞吐能力等方面。

服务器的分类有很多种，按其规模可划分为大型服务器、中型服务器、小型服务器、瘦服务器等。瘦服务器指的是剥去了附加部件，只包含执行一些专用任务所需功能的简化了的服务器。在本文中，综合了将嵌入式系统接入 Internet 的各种方法，提出了瘦服务器的概念。一方面，网络功能的实现只在瘦服务器上完成，不需要占用嵌入式系统的存储容量；另一方面，嵌入式系统一般都带有数据通信的接口，可以按照 RS-232、RS-485、CAN 等协议格式发送数据。瘦服务器只要设置相应的接口，便可以和嵌入式系统相互交换信息。这样，便可以将嵌入式系统作为一个独立的模块分离出来，网络功能也和具体的嵌入式系统脱离开了关系，便于设计和开发。

瘦服务器从功能上来说还是一个服务器，它之所以能够从软件上和硬件上“瘦”下来，是因为瘦服务器不象一般的服务器那样对操作系统软件要求使用 Windows NT、UNIX、Windows 2000 等大型的操作系统。这样的多任务操作系统必然占用大量的存储容量，同时要求 CPU 有很快的运算速度，内存比较大，硬盘有很大的容量等，对硬件的要求也比较高。而瘦服务器因为资源的原因无法使用多任务的操作系统，所以采用顺序执行和硬件中断的方式，这样一来，对 CPU 速度、内存容量、硬盘容量的要求大大降低。

本文就是要利用嵌入式 Internet 技术实现一个瘦服务器并将它应用于实践。

1.2 国内外研究现状

以 Internet 为代表的计算机网络的迅速发展及相关技术的日益完善，突破了

传统通信方式的时空限制和地域障碍，使更大范围内的通信变得十分容易。Internet 拥有的硬件和软件资源正在越来越多的领域中得到应用，比如电子商务、网上教学、远程医疗、远程数据采集与控制、高档测量仪器设备资源的远程实时调用、远程设备故障诊断等等。与此同时，高性能、高可靠性、低成本的网关、路由器、中继器及网络接口芯片等网络互联设备的不断进步，又方便了 Internet、不同类型测控网络、企业网络间的互联。利用现有 Internet 资源而无需建立专门的拓扑网络，使组建测控网络、企业内部网络以及它们与 Internet 的互联都十分方便，这就为测控网络的普遍建立和广泛应用铺平了道路。

把 TCP/IP 协议作为一种嵌入式的应用，嵌入现场智能仪器的 ROM 中，形成瘦服务器，使信号的收、发都以 TCP/IP 方式进行，如此，测控系统在数据采集、信息发布、系统集成等方面都以企业内部网络为依托，将测控网和企业内部网及 Internet 互联，便于实现测控网和信息网的统一。在这样构成的测控网络中，传统仪器设备充当着网络中独立节点的角色，信息可跨越网络传输至所及的任何领域，实时、动态（包括远程）的在线测控成为现实。将这样的测量技术与过去的测控、测试技术相比不难发现，今天的测控能节约大量现场布线、扩大测控系统所及地域范围。使系统扩充和维护都极大便利的原因，就是因为在这种现代测量任务的执行和完成过程中，网络发挥了不可替代的关键作用，即网络实实在在地介入了现代测量与测控的全过程。

单片机系统实现连接 Internet 的技术难点在于：如何利用单片机系统自身的资源对信息数据进行 TCP/IP 协议处理，使之变成可以在 Internet 上传输的 IP 数据包，从解决这一技术问题出发，目前出现了以下几种方案：

1．采用 EMIT 技术[5]

EMIT（Embedded Micro Internetworking Technology）是由美国Emware 公司推出的一套成熟、完善的实现单片机系统与 Internet 相连的解决方案，EMIT 技术体系结构如图 1-1 所示。

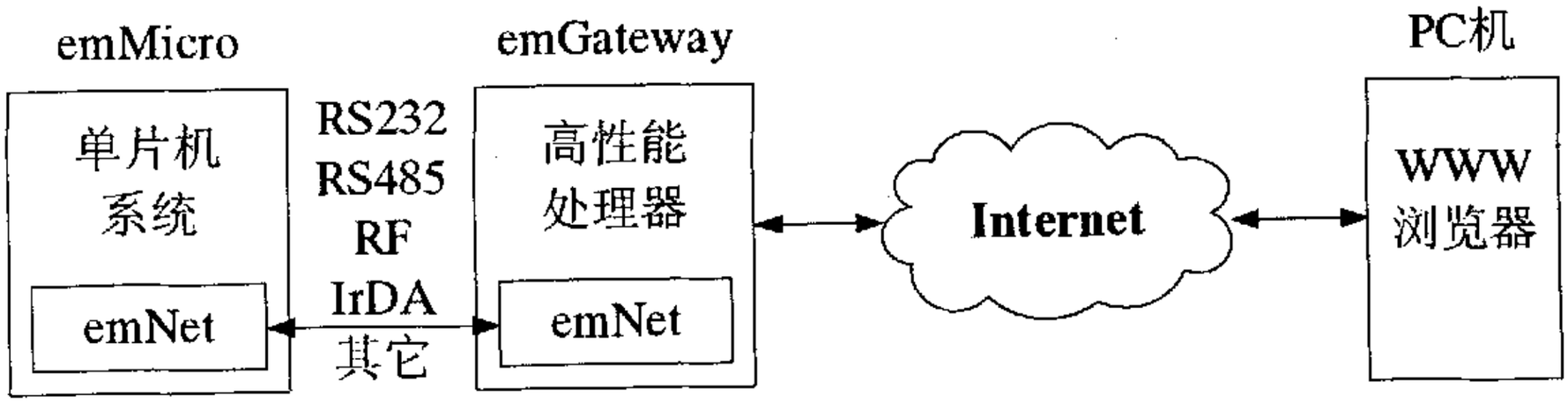

图 1-1 EMIT 技术体系结构

EMIT 采用桌面机或高性能的嵌入式处理器作为网关，称为 emGateway，上面支持 TCP/IP 协议并运行 HTTP 服务程序，形成一个用户可以通过网络浏览器进行远程访问的服务器。emGateway 通过 RS-232、RS-485、CAN、红外、射频等总线与多个嵌入式设备相连接，每个嵌入式设备的应用程序中包含一个被称为 emMicro 的独立通信任务，该任务负责检测设备中预先定义的各个变量，并将结果反馈到 emGateway 中；同时它还可以解释 emGateway 的命令，修改设备中的变量或者进行某种控制。EMIT 解决方案中还包括 EMIT 应用接口服务技术，它提供一系列的开发工具，如 C++、Java 以及 Active X 等。用户可以根据需要开发客户端的浏览界面。

EMIT 技术经过多年的发展已经在工业设备的网络化中得到了广泛的应用，得到了包括 Motorola、AT&T、Hitachi 等多家 IT 公司的支持。但该技术也存在着以下的缺点：EMIT 起步价位很高，emGateway 网关的使用需要具有许可证，要缴纳相应的版税；在多个嵌入式设备分散的情况下，网关和设备间的专用网络布线极为不便，通信的距离、速度都受到一定的限制。

2. 采用 Webchip 芯片实现单片机系统网络化[6]

Webchip 是我国武汉力源电子股份有限公司推出的可实现单片机系统与 Internet 连接的接口芯片。它是独立于各种微控制器的专用网络芯片，在它的片内驻留着网络协议解释和网络协议编译程序模块，通过标准的 SPI 串口与嵌入式系统中的微控制器相连接。微控制器可以通过 Webchip 接收并执行经由 Internet 远程传来的命令，或将一些数据交给 Webchip 发送出去。单片机系统采用 Webchip 接入 Internet 的网络，还必须有一个支持复杂 Internet 协议并能提供 HTTP 服务的类似于 emGateway 的网关。事实上，Webchip 芯片，只是把在 EMIT 技术中由微型网络服务器 emMicro 完成的网络功能独立于原来的微控制器，把软件调用变成了硬件多控制器之间的通信。因此它只是实现了 EMIT 技术的一小部分功能，距离用单个片上系统完成智能装置网络化的目标还相差甚远。

3. 采用硬件协议栈芯片

硬件协议栈芯片是一个独立于各种微控制器的具有上网功能的专用芯片，通过标准的输入输出接口，可以和大多数的微控制器相连。这些微控制器可以通过硬件协议栈芯片执行由 Internet 远程传来的命令，或是将数据交给硬件协议栈芯片通过 Internet 发送出去。典型的硬件协议栈芯片有日本 Seiko 公司以 iReady 芯核为基础，开发的 S7600 芯片等。硬件协议栈芯片与 Webchip 芯片的最根本

的区别在于，嵌入式系统连接 Internet 不需要再通过额外的网关。图 1-2 显示了一种典型的基于 S7600 芯片的嵌入式 Internet 的方案[5]。

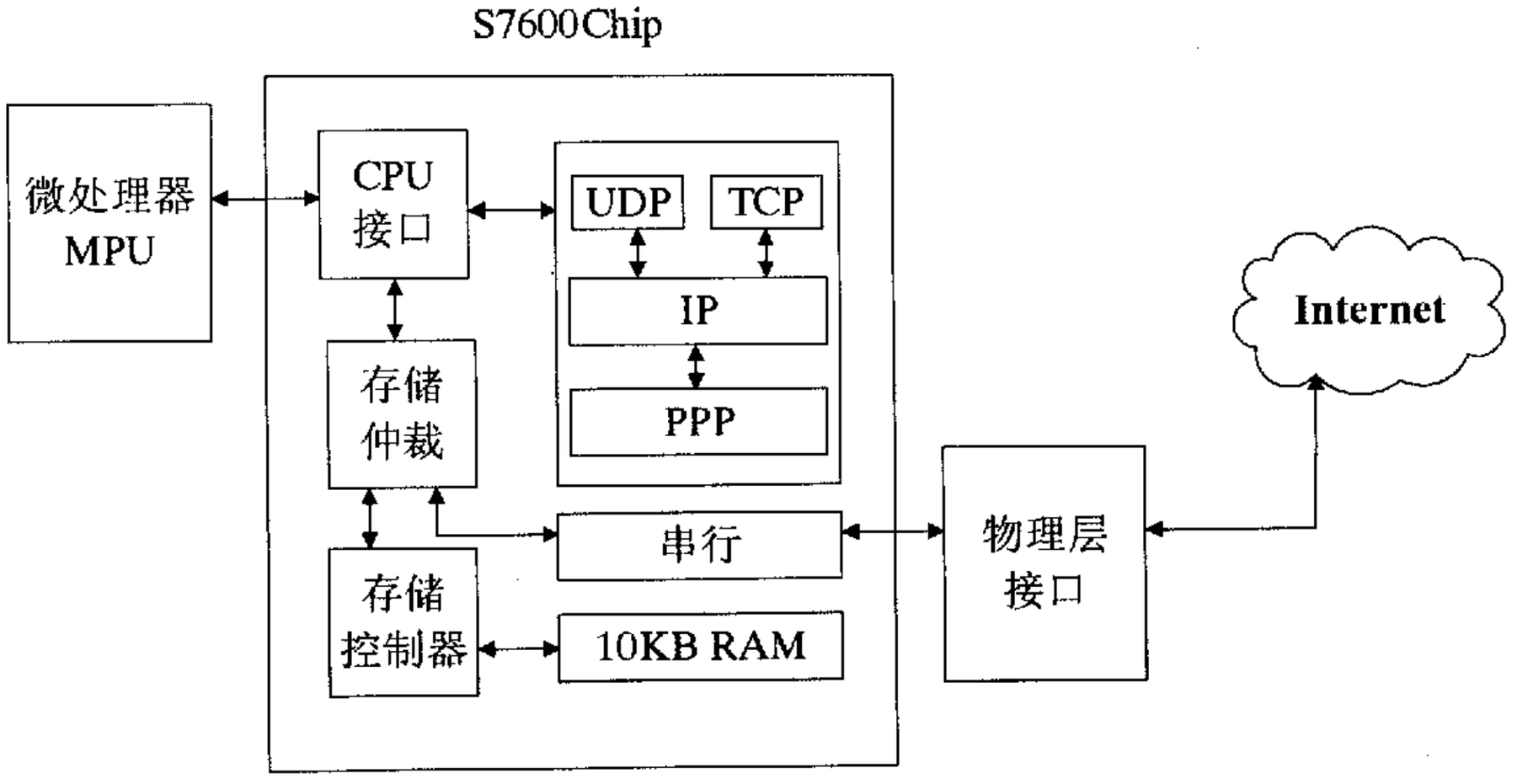

图 1-2 基于 S7600 的一种方案

当使用硬件协议栈芯片开发具有网络功能的单片机系统时，只需要增加一段和协议栈通信的接口程序即可，因此，系统软件开发难度较小，周期也较短。但是由于加入协议栈芯片以及与之配套的网络连接层设备（如嵌入式 Modem、以太网控制器等），增加了系统的硬件成本。

4．采用支持 Internet 的嵌入式操作系统

嵌入式操作系统是指运行在嵌入式平台上，内核小、具有高度的模块化和扩展性；具备文件和目录管理、设备支持、多任务、图形窗口以及用户界面等功能；具有大量的应用程序接口（API），能够适应系统对功能、可靠性、成本、体积和功耗严格要求的系统软件。由于 Internet 技术的发展，嵌入式操作系统已逐渐向网络化的方向发展。目前市场上已有许多这样的嵌入式操作系统，如 uCLinux、RTLinux、VxWorks、Nucleus Plus 以及 VRTXsa 等[7]，它们通过内核自身或是附加的网络组件提供对 TCP/IP 协议的支持。基于这些操作系统的嵌入式设备，由于操作系统自身的要求大都采用高档的 32 位处理器，如 x86、PPC、ARM、ColdFie 等，同时拥有较大容量的 ROM 和 RAM，因此设备能实现多种复杂的网络功能。但这种方案存在如下缺点：高档的 32 位处理器价格较贵，开发周期较长；需要购买昂贵的嵌入式操作系统、网络组件以及配套的开发软件，

对开发人员的开发能力、经验要求较高。

5．采用 8 位或 16 位高性能的微处理器 + 精简 TCP/IP 协议栈

根据嵌入式网络产品应用的特点，将标准的 TCP/IP 协议栈做较大幅度的简化，根据需要保留其中基本的部分，这样就可以大幅度减少对于系统资源的需求，从而可以在低成本、小内存的系统中实现 Internet 连接的功能，这样的方案的优点是廉价，便于广泛应用。该方案的重点和难点是实现精简的 TCP/IP 协议栈。以往人们认为基于 8 位或 16 位微处理器的单片机系统由于在处理能力、ROM 和 RAM 空间上的限制，不能实现 TCP/IP 网络协议栈等这样复杂的功能[8][9]。但是，近年来随着市场对超微型嵌入式应用技术的不断增长，以及半导体技术和系统设计方法的进步，在一个硅片上实现一个过去以为复杂的系统的时代已经来临，并深刻地影响着传统的集成电路产业[4]。不断发展的具有高速处理能力的智能化嵌入式芯片，使得单片机系统支持 TCP/IP 协议成为可能。

在综合考虑了以上的方案后，本文采用的是通过精简嵌入式 TCP/IP 协议栈实现嵌入式 Internet 的方案。

1.3 本课题的主要任务和目标

综合前文所述可知，随着电于设备智能化要求的提高和各种功能强大的微处理器的推出，单片机系统逐渐成为许多电子设备不可缺少的一部分，并处在前所未有的发展期；而互联网的概念深入人心更为单片机系统与互联网的结合提出了要求。计算机、通信网络在 Internet 广泛使用的推动下得到迅猛的发展。可以预见在未来十年中，这种发展速度不仅不会减慢，反而会加快并向人们活动的各个领域渗透。Internet 这种广泛应用基本推动力是源于现代人类活动的高度流动性，以及在其过程中对信息远程获取（Remote Access）的需求的不断增高。因此可以预见，对电子设备进行 Web 化，将成为信息时代即将来到的新浪潮。电子设备的 Web 化，实质上是嵌入式 Internet 技术的应用，它将把互联网方式更加自然、深入地带进我们的日常生活[10]。

由单片机系统、嵌入式网络的应用环境决定了嵌入式 TCP/IP 协议栈通常应用于特殊的、专用的领域，不可能像标准的 TCP/IP 协议栈一样提供完整的协议体系，往往是根据具体的应用提供不同的协议模块[11]。因此，嵌入式 TCP/IP 协议栈区别于标准的 TCP/IP 协议栈的最突出的特点有如下三点。

1．很好的可裁剪性

由于嵌入式应用的要求千差万别，各种嵌入式应用对系统的要求不尽相同，并且在嵌入式应用中对产品的成本、价格比较敏感，存储器的容量往往都是比较有限的，因此必须根据嵌入式网络产品的具体功能，对完整的 TCP/IP 协议栈功能进行裁剪，特别是对应用协议提供可裁剪性，以满足用户的需求。

2．很强的可移植性

由嵌入式应用的多样性决定了嵌入式应用平台也是变化多端的。因此，在我们开发网络协议栈软件的过程中，保证软件的可移植性是非常重要的。这样，在对嵌入式产品进行软、硬件升级的过程中除了与硬件直接相关的部分代码需要重新编写以外，不必再对上层协议进行大的修改。

3．代码精简

嵌入式 TCP/IP 协议栈是标准 TCP/IP 协议栈的子集，只需要实现基本的、必要的功能，使生成的二进制代码尽量精简，这对嵌入式网络产品降低开发难度、提高系统处理能力、节省有限的 ROM 和 RAM 空间是有着重要的意义的。嵌入式 TCP/IP 协议栈的这些特点也将成为本课题研究工作的目标与要求[14]。

根据以上这些特点，本文精简了 TCP/IP 协议，实现的瘦服务器是采用 51 处理器的单片机系统，这些系统广泛应用在工业控制、通信设备、家用电器等领域中。这些为数众多的设备接入 Internet 后，将可实现设备的远程控制、管理和升级等功能，改变以往单独、孤立的存在方式，进入一种开放、互连的方式[15]。

第 2 章 瘦服务器应用系统简介

2.1 瘦服务器应用系统概述

本课题实现的瘦服务器应用系统采用客户/服务器（Client/Server）模式。客户/服务器模型又称为主从式模型。在一个信息处理系统中，用于提供数据和服务的计算机称为服务器（server）；向服务器请求服务和数据的计算机称为客户机（client）。这样的系统模式称为客户/服务器模式。

客户端向网络上某一服务器发出请求，该请求被 TCP/IP 网络传递到所需的服务器上，服务器对该请求做出响应。同样，通过网络该响应被传回客户。在客户/服务器模型中，服务器程序作为一个守护进程，随系统启动而启动。当无请求时，服务器处于等待状态；当请求到达时，服务器立即处理请求并做出响应，而自己本身则回到等待状态。在这种方式下，服务器可以很好地处理网络上客户随机的请求。

通信和资源共享是计算机网络的两大功能，客户/服务器模型是这两方面在客观现实中的完美体现。

首先，网络通信是一种完全异步的通信，通信的发生完全是随机的，相互通信进程之间不存在着父子关系，也不共享内存缓存，因此需要一种机制，为准备通信的进程之间建立联系，为两者的数据交换提供同步。客户/服务器模型很好地解决了这个问题。按照 C/S 模型，每次网络通信均由随机运行的客户进程发起，服务器进程从启动开始就一直处于等待状态。这样，就保证了服务器随时对客户请求做出响应。另外，客户与服务器之间的请求应答模式为相互通信间的数据传输同步提供有力支持。

其次，从资源角度看，Internet 上大量客观存在着资源分布和运算能力不均等现象：小到一个物理网络，某些主机拥有大容量的外存，某些主机则只有少量甚至没有外存；大到整个 Internet，少数网点拥有超级运算能力，大量网点只有 PC 机。同时，Internet 上还存在人为的不均现象(主要指信息资源方面)，如域名信息等。这些信息往往以数据库形式存在于少数特权主机当中，供局部或全局访问。总之，资源不均等的现象是不可避免，而客户/服务器模型很好地适应了这种现象。

正是基于以上两点，在实际应用中，我们将实现的瘦服务器应用在印刷机的套色控制中，组成C/S模式的测控网络。瘦服务器的结构示意图如图2-1所示[38]。瘦服务器通过RJ45接口接入局域网，再通过局域网连上Internet。现场数据通过瘦服务器的传感器接口、I/O接口、RS232端口采集到Cygnal单片机上。这时，用户使用一台连接上Internet的电脑作为客户机就可以通过瘦服务器实现印刷机的远程控制。只是在客户机上需要运行一个应用程序。

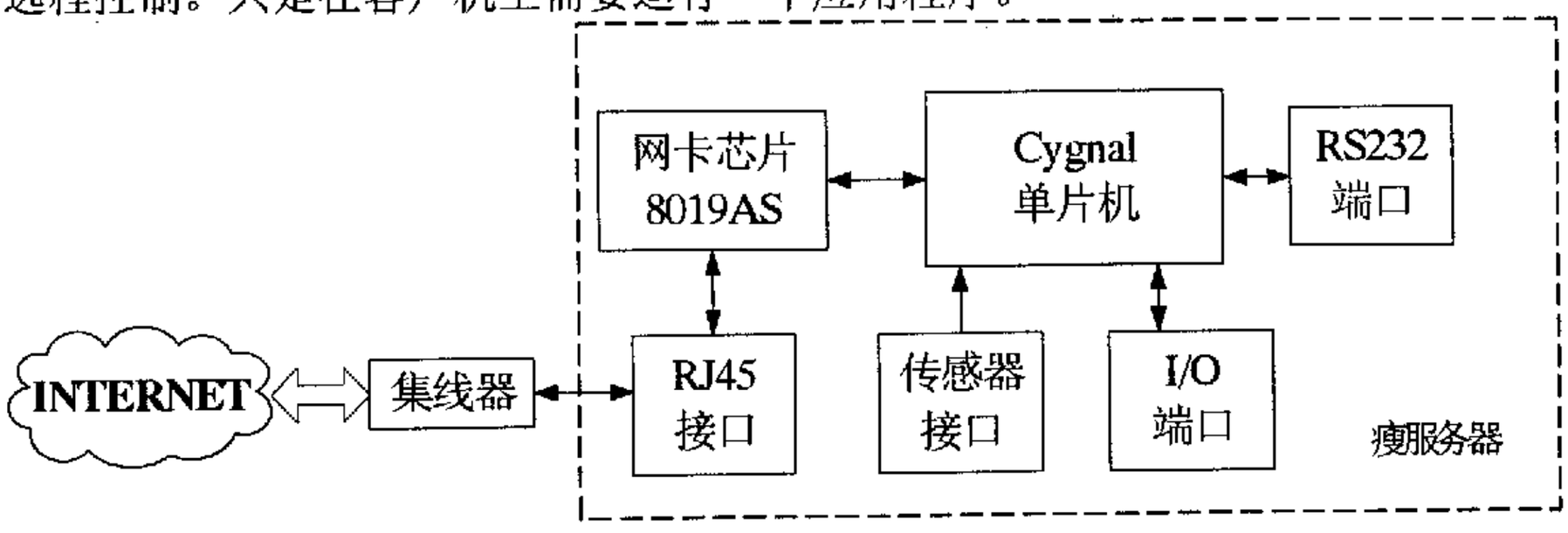

图 2-1 瘦服务器结构示意图

一般地，一台大型印刷机有 6/9/12 个印刷控制单元，每个印刷控制单元由实现网络功能的瘦服务器和印刷控制模块两部分组成，负责印刷一种颜色。印刷时，依次印刷不同的颜色以组成最后的图案。如果每种颜色印刷准确，也就是说颜色套准，那么图案清晰美观，界限分明；如果颜色没有套准，则会出现图案重叠或者模糊。本文设计的印刷控制系统由工业 PC 机（在系统中作为客户机）、17 寸彩色显示器（触摸屏）、若干个瘦服务器、光电编码器、精密电源、光电眼、执行机构和控制软件组成。各部件（控制、电源、信号等）通过集线器相连形成局域网，再连接上 Internet。各主要部件的功能、用途如下：

- 工业 PC 机——是本系统的客户机，运行客户端程序。用于接收、发送和处理各个瘦服务器、光电编码器、光电眼等传输的信息。
- 印刷控制单元——印刷控制单元包括瘦服务器和印刷控制模块。印刷控制模块在接收光电眼检测的色标信号后，进行处理和误差计算，并根据计算结果，驱动执行机构动作，完成自动套色控制。同时瘦服务器将误差和修正结果报告给客户端。控制单元数与光电眼数是一一对应的。
- 光电编码器——是用来检测版辊位置的，一般安装在印刷机主轴上。和版辊同步转动，一圈输出 1000 个脉冲。

- 光电眼——也叫色标传感器。用于检测印刷色标位置。

在本应用系统中，瘦服务器既负责接收客户端用户发送的操作指令，控制印刷动作准确无误地完成；又通过传感器接口获得光电眼采集的数据返回给客户端，以便用户能根据实际的印刷情况及时调整和完成实时控制。最后组成的控制系统如图2-2所示。

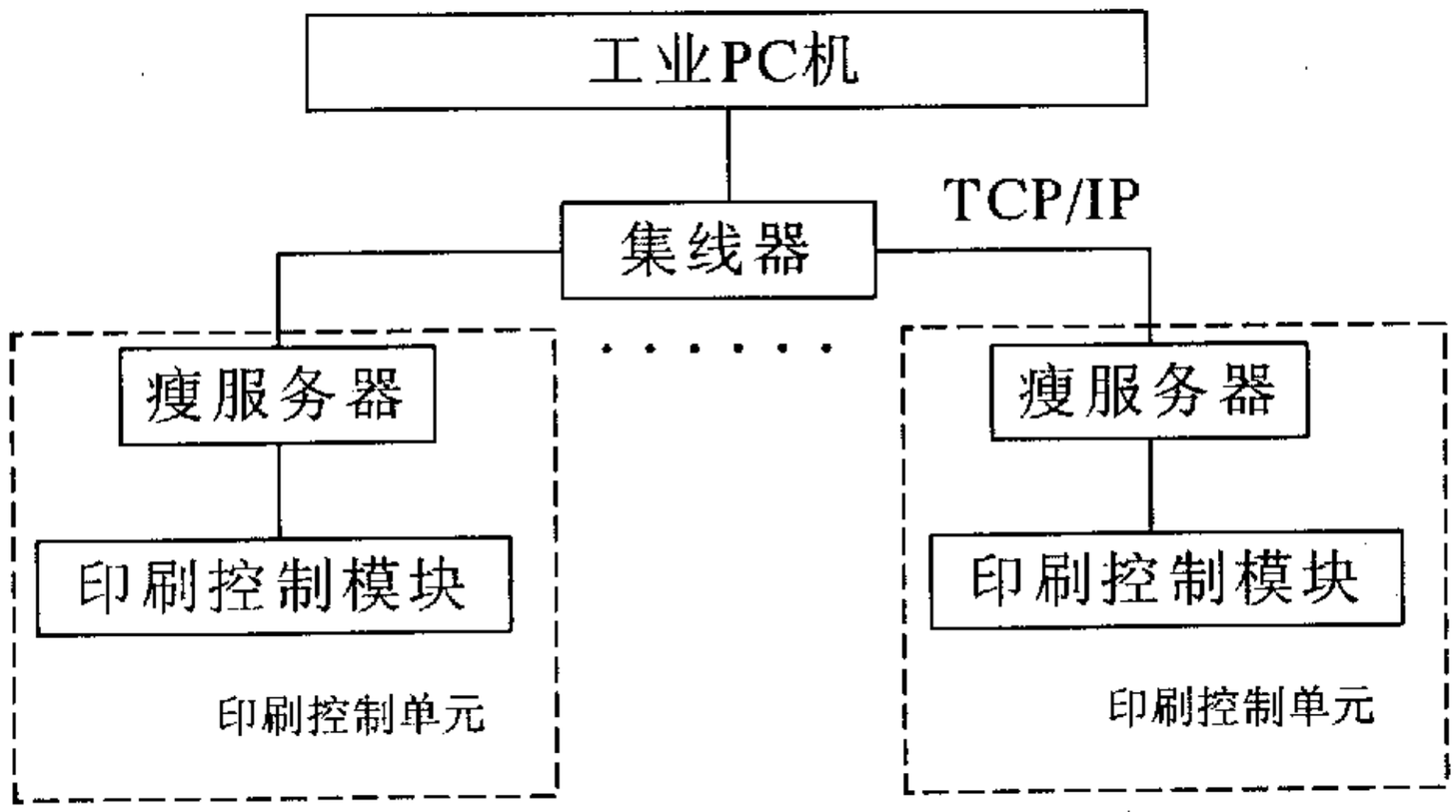

图 2-2 套色控制系统结构图

在 C/S 模型中，客户与服务器处于不平等的地位，服务器拥有客户所不具备的硬、软件资源和运算能力，服务器提供服务，客户请求服务。同时，客户/服务器模型对于网络应用程序的开发具有指导意义。也就是说，应用程序的开发由此可以分为两部分在客户端的和服务器两端进行，而且大部分地集中在服务器端。这充分利用了服务器的资源，不仅可以大大节约应用程序开发的精力，同时也减轻了客户的压力，有利于应用程序的推广使用。

2.2 应用系统中的客户端

在应用系统中，以一台工业PC机作为客户机，当需要进行远程测量和控制时，客户机向瘦服务器发出请求，瘦服务器收到请求后，启动数据采集程序，并将采集到的数据通过网络传输到客户机上，客户机对这些数据进行分析、处理和显示，然后用户根据分析的结果做出相应的控制，同样通过网络发送回瘦服务器，最终完成对印刷机的远程控制。

本应用中，客户端需要运行一个应用程序。因为是应用在工业上，并且客

户端还会使用触摸屏来简化控制操作，所以该程序要求能提供友好的人机界面，最好设计成一些工业上常用的仪器仪表的界面，这样最终的用户就能熟悉地操作和使用。同时，该程序还必须实现Internet网络功能，既能将控制数据按网络数据报的格式进行打包，又能将收到的网络数据正确解包获得有用信息。基于上述各方面的考虑，最后我们决定采用美国国家仪器公司（National Instruments，NI）的LabVIEW软件来编写这个应用程序。

下面将对虚拟仪器及其编写软件LabVIEW做简单的介绍。

2.2.1 虚拟仪器概念及 LabVIEW

虚拟仪器（Virtual Instrument，VI）是在通用计算机平台上，用户根据自己的需求来定义和设计测试功能的仪器系统。也就是说虚拟仪器是由用户利用一些基本硬件及软件编程技术组成的各种各样的仪器系统[42]。虚拟仪器技术是测控技术领域出现的一项突破性进展，是当今计算机辅助测试（CAT）的一项重要技术。

虚拟仪器一般由系统硬件、系统软件以及为进一步扩展功能所留出的软硬件接口等组成。系统硬件的核心部分可以是微型计算机、单片机或由其他类型计算机组成的混合网络系统，其余则可包括A/D、D/A 、量程自动切换、功率驱动、滤波及其他一些处理电路。系统软件则可包括数据采集、信号分析与处理、系统仿真、控制算法等模块，这些软件资源一般都利用面向对象的程序设计方法开发，因此用户可以方便地在此基础上进行二次开发，充分体现了虚拟仪器的“软件就是仪器(The software is the instrument)”的思想[43]。

通常在编制虚拟仪器的软件时可以采用两种编程方法：一种是传统的编程方法，采用高级语言，如 VC++、VB 等编写虚拟仪器的软件；另一种是采用现在流行的图形化编程方法，如用 NI 公司的 LabVIEW（Laboratory Virtual Instrument Engineering Workbench，实验室虚拟仪器工作平台）或 HP 公司的 VEE 等编程。采用图形化编程的优势是软件开发周期短、编程容易，特别适合不具有专业编程水平的工程技术人员使用。

LabVIEW是一个基于图形化编程语言（G语言）的虚拟仪器开发环境，它提供了一种全新的编程方法，即对称之为“虚拟仪器” 的软件对象进行图形化的操作组合。利用LabVIEW，可以通过交互式的图形化前面板来控制系统，例如：可以对几千种硬件设备（GPIB、VXI、PXI、RS-232、RS-485、PLC、插入式数

据采集卡等）进行数据采集；可以通过网络、交互应用通讯和结构化查询语言(SQL) 等方式与其他数据源相连；可以利用其功能强大的数据分析程序对原始数据进行分析，得到有意义的结果并加以显示输出；可以创建执行速度较快的32位编译程序，用来实现数据采集等测试任务。LabVIEW带有大量数据采集、分析、显示、存储的函数库以及众多的程序开发工具，还通过动态链接库DLL、共享库函数、Active X等提供了大量的外部代码接口。在调试方面，具有设置断点、单步运行等有力的功能。

2.2.2 虚拟仪器网络化

当今时代，自动测试仪器系统的发展方向是：智能化、标准化和网络化；计算机网络化的迅速普及和深入发展，为仪器的网络化奠定了基础。将网络技术与虚拟仪器相结合，构建网络化虚拟仪器系统，是虚拟仪器系统发展的方向之一。

LabVIEW作为一个优秀的虚拟仪器开发平台，不仅充分体现了“软件就是仪器”的虚拟仪器概念，而且，LabVIEW为构建基于计算机网络的测试系统，提供了多种功能强大的工具，通过LabVIEW的网络功能，可以很方便地实现虚拟仪器的“网络化”扩展。

虚拟仪器在以下几个方面具有传统仪器无可比拟的特点[44]：

1．虚拟仪器的功能、性能、指标可由用户定义，彻底打破了传统仪器一经设计、制造完成后，其功能、性能、指标不可改变的封闭性、单一性。

2．可以将多种仪器的功能、性能、指标等以软件的形式集成在一个“功能软件库”——虚拟仪器库内，通过它们的不同组合以及与各种不同类型的硬件接口搭配，使得在一台个人计算机就可实现各种仪器的不同功能，大大提高了仪器功能的灵活性，甚至可以进行非常复杂性的测试工作。

3．由于计算机具有强大的图形界面功能和数据运算功能，因此虚拟仪器的操作简单直观，数据分析及数据处理、结果与图形曲线的显示功能也非常强大。

4．同一系统中的仪器之间可以通过网络进行数据交换，实现资源共享。

5．开发周期短、成本低、维护方便，易于应用新理论、新算法和新技术，实现仪器的换代升级。

正是由于这些原因，所以在客户端利用LabVIEW编写应用程序。但本文主要介绍的是系统中瘦服务器的实现与应用，故应用程序的设计与实现部分在此

不再赘述。

2.3 应用系统中的瘦服务器端

在本应用系统的运行中，瘦服务器始终处于等待状态，直到客户向它发出请求。当它接收到客户端的请求后，要根据请求做出响应。一方面，瘦服务器要负责接收客户端通过网络发送过来的控制指令；另一方面，瘦服务器还要负责将各种现场采集到的数据按要求通过网络返回给发出请求的客户机。所以，要实现瘦服务器实际上就是要在一个特定的嵌入式系统（即瘦服务器硬件）中实现精简的TCP/IP协议。

瘦服务器的硬件框图参见图2-1，其中主要的功能芯片是Cygnal单片机和以太网接口芯片RTL8019AS。

瘦服务器软件程序的总体数据流图如图 2-3 所示。发送数据时，印刷机的控制程序将采集到的现场数据交给 UDP 协议模块处理，UDP 模块将其首部和数据封装成 UDP 数据报。然后将封装好的 UDP 数据报交给 IP 协议模块，IP 模块在 UDP 数据报上添加 IP 首部，并封装成 IP 数据报，然后根据路由表为 IP 数据报确定路由，如果找不到相应路由，则向 ICMP 协议发送出错报文，由 ICMP 协议模块进行处理；找到了路由则将数据报发送到网络接口层，网络接口层利用 ARP 协议找到目的的 IP 地址对应的物理地址，然后封装成以太网帧，由网卡驱动程序将以太网帧发送出去。

接收数据时，由网卡驱动程序负责接收数据，然后由中断处理程序唤醒数据接收任务，由数据接收任务将接收到的数据帧交给网络接口层。网络接口层取出帧头，判断接收数据的类型，如果是ARP报文，则将ARP报文交给ARP协议模块处理；如果是IP数据报，则将IP数据包交给IP协议模块处理，IP协议模块取出IP首部信息，然后根据数据报的类型，将报文交给相应的协议模块（UDP模块或ICMP模块）处理；UDP模块收到报文后，取出首部进行处理，并将用户数据交给印刷机的控制程序[28][29]。

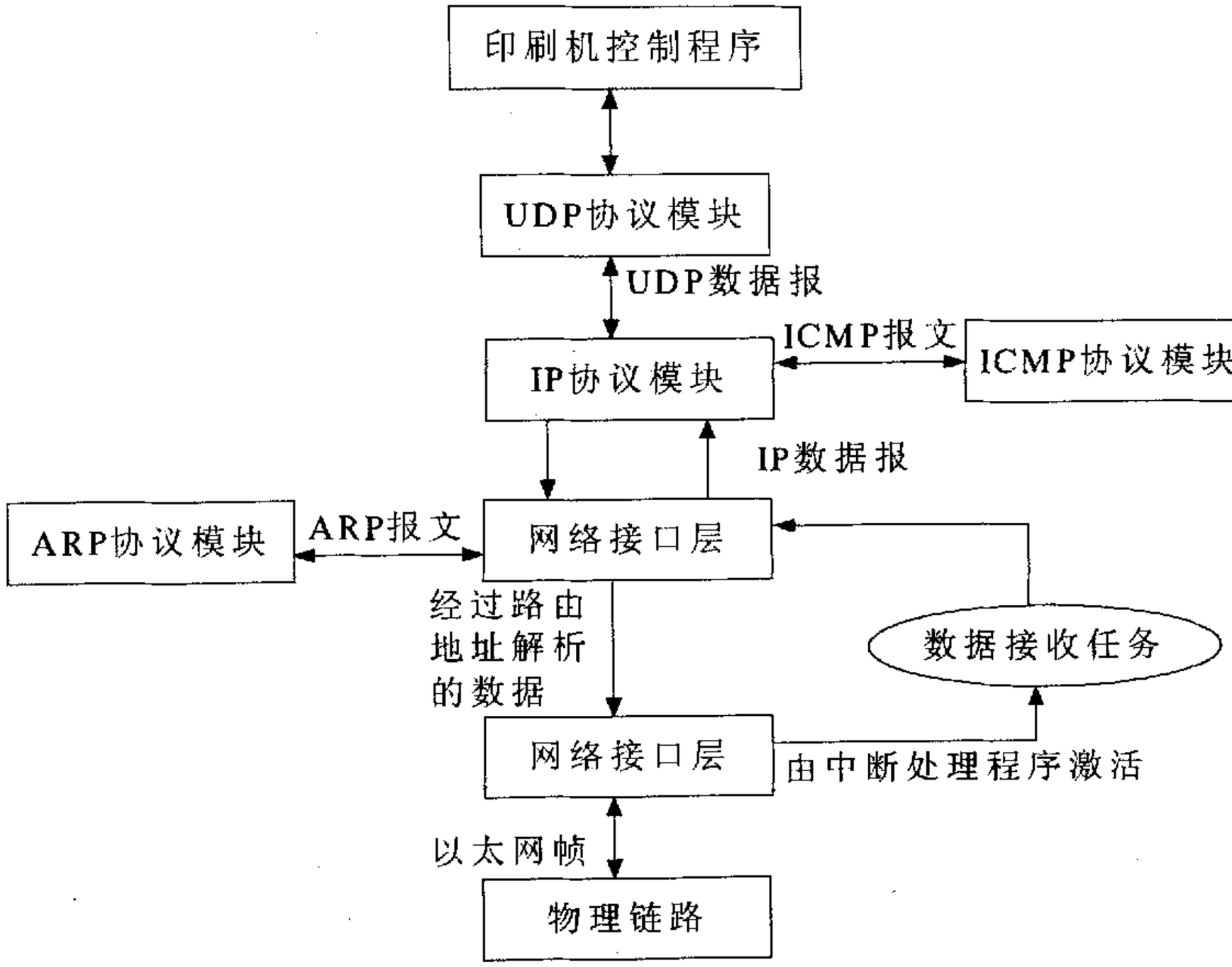

图 2-3 瘦服务器的总体数据流图

瘦服务器硬件及软件的实现将在下一章具体介绍。

第 3 章 瘦服务器的设计与实现

当人们谈起互联网时，常常会谈到TCP/IP。实际上TCP/IP是用于计算机通信的一组协议，我们通常称它为TCP/IP协议族。它包括TCP、IP、UDP、ICMP、RIP、TELNETFTP、SMTP、ARP、FTP等许多协议，是70年代中期美国国防部为其ARPANET广域网开发的网络体系结构和协议标准，以它为基础组建的Internet是目前国际上规模最大的计算机网络，正因为Internet的广泛使用，使得TCP/IP成了事实上的标准[21]。

通常的TCP/IP五层参考模型如图3-1所示，即是建立在第五层（硬件层）上的四个软件层。从这个层次结构上，我们很容易清楚地看出数据在Internet上的不同主机间的流动情况[22]。

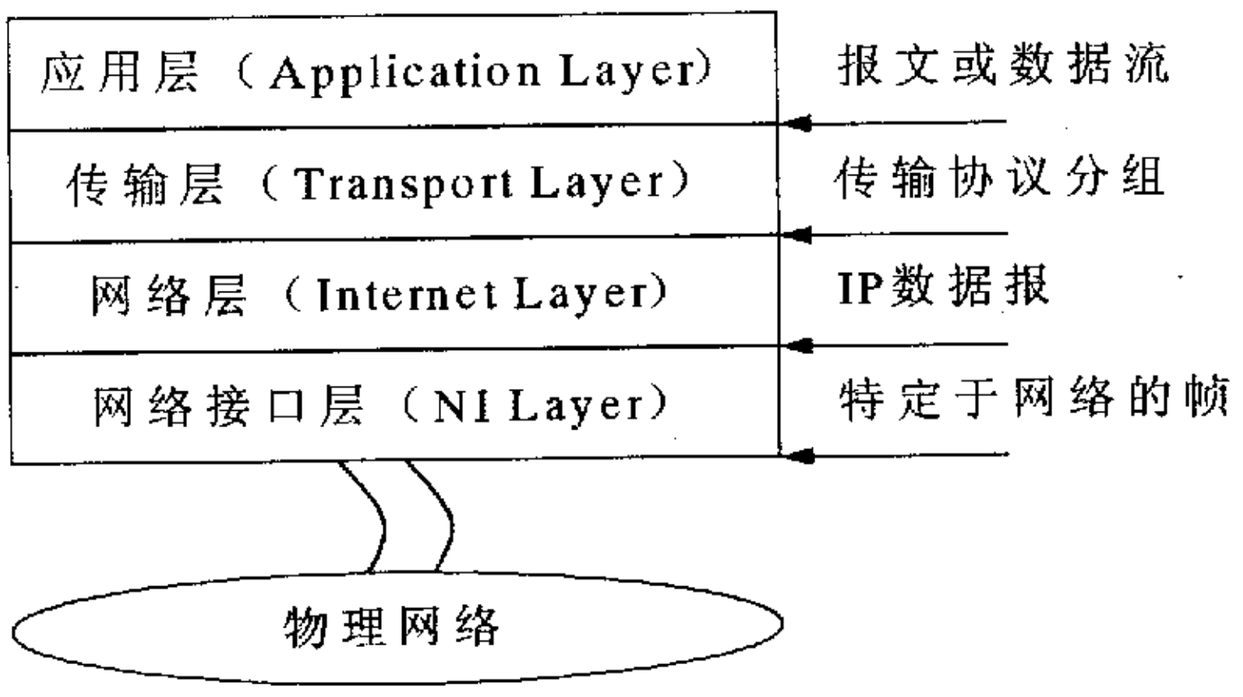

图 3-1 TCP/IP 五层参考模型

其中，网络接口层又称链路层或数据链路层，这是TCP/IP软件的最底层，负责接收IP数据报并通过网络发送之，或者从网络上接收物理帧，抽取出IP数据报，交给IP层。通常包括操作系统中的设备驱动程序和对应的网络接口卡。它们一起处理与电缆（或其它任何传输媒介）的物理接口细节。

网络层又称互联网层或网间网层，处理分组在网络中的活动，例如分组的选路。即负责相邻计算机之间的通信。其功能包括三方面。①处理来自传输层的分组发送请求，收到请求后，将分组装入IP数据报，填充报头，选择去往信宿机的路径，然后将数据报发往适当的网络接口。②处理输入数据报：首先检查其合法性，然后进行寻径——假如该数据报已到达信宿机，则去掉报头，将剩

下部分交给适当的传输协议；假如该数据报尚未到达信宿，则转发该数据报。③处理路径、流控、拥塞等问题。

传输层主要为两台主机上的应用程序间提供端到端的通信。其功能包括：①格式化信息流；②提供可靠传输。这层包含两个互不相同的协议：TCP（传输控制协议）和UDP（用户数据报协议）[23]。

应用层向用户提供一组常用的应用程序，负责处理特定的应用程序细节。比如电子邮件、文件传输访问、远程登录等。远程登录TELNET，使用TELNET协议提供在网络其它主机上注册的接口；文件传输访问FTP，使用FTP协议来提供网络内机器间的文件拷贝功能。

一个完整的各层间数据传输流程如图3-2示。

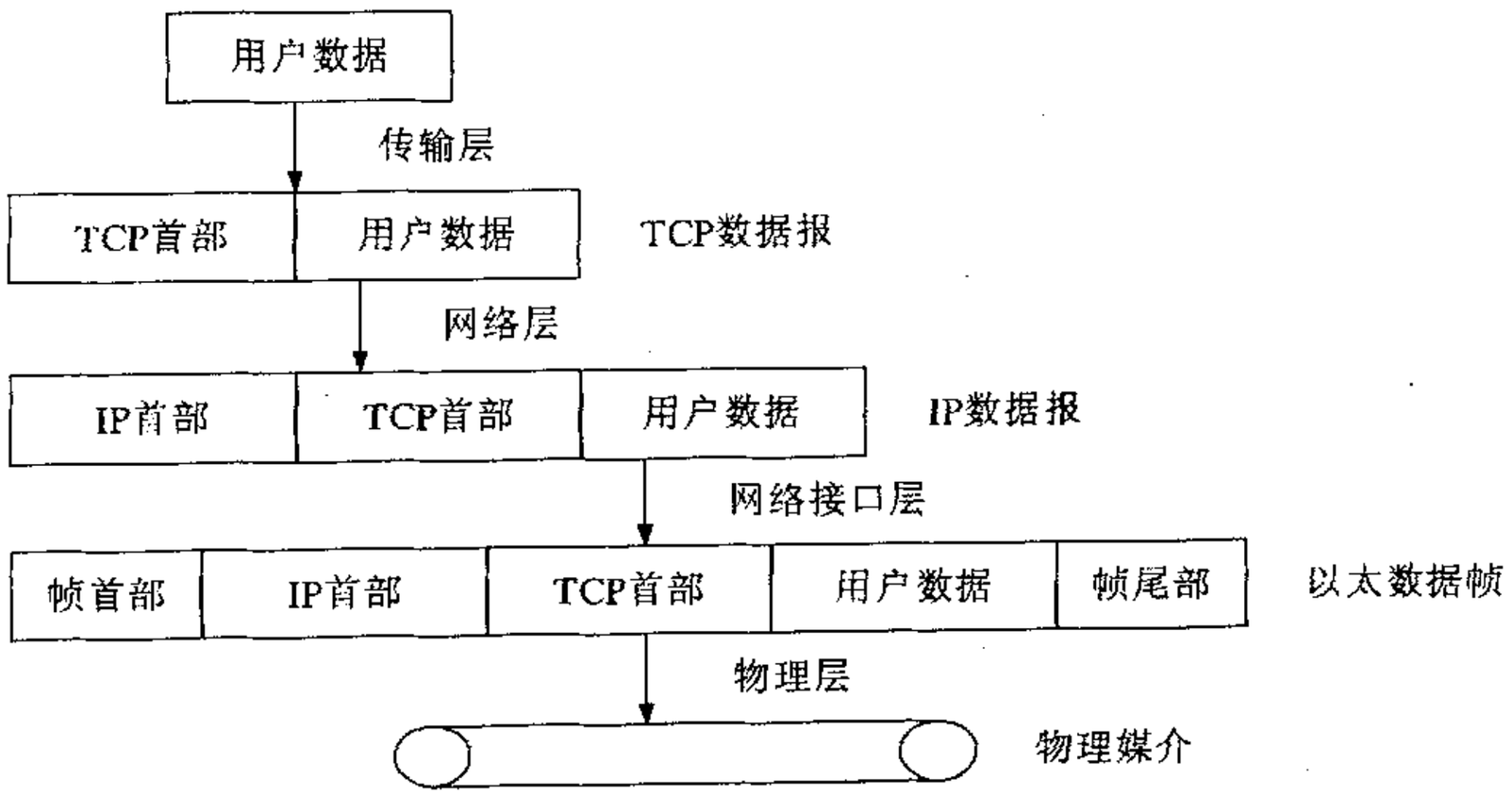

图 3-2 TCP/IP 各层中数据流动示意

TCP/IP协议族中，ARP协议是网络接口层协议，实现地址解析功能；IP是网络层协议，实现的是不可靠无连接的数据包服务；TCP和UDP都是建立在IP基础上的传输层协议。TCP是基于可靠连接的协议，UDP是基于不可靠无连接的协议。

在本系统中，一方面考虑是实现印刷机套色系统的网络控制，系统是应用在高可靠性、低延迟的局域网上，而且印刷机的机械速度远远低于 CPU 的运算速度，对系统的实时性要求不高，需要网络传输的数据量也不大；另一方面，由于瘦服务器是基于嵌入式 Internet 技术的，它的硬件实际上是一个单片机的嵌入式系统，CPU 的速度、内存的容量都有限，所以在瘦服务器中的传输层，本论文选择实现 UDP 协议而不是 TCP 协议。另外，在网络层和网络接口层，本论

文分别实现了IP、ICMP、以太网控制协议以及网卡芯片的驱动程序。

本章主要介绍了基于嵌入式Internet技术的瘦服务器软、硬件的设计与实现。其中，软件的编程是实现本方案的关键，网络通信中的数据封装、解析都是由单片机软件实现的。按照功能，软件实现主要包括两部分：其一是网卡芯片RTL8019AS的驱动程序，主要包括：RTL8019AS的初始化、数据帧的发送、接收过程；其二是单片机上精简TCP/IP协议栈的实现、对数据的逐层打包、封帧、传送等流程。

3.1 瘦服务器的硬件组成及关键芯片的驱动程序

3.1.1 瘦服务器硬件框图

瘦服务器的硬件电路主要由以下几个部分组成：Cygnal 单片机C8051F020、128K 的 SRAM UT62L1024、SPI 串行方式 8M 位的 Flash 存储器AT45DB081 和 10M 以太网芯片 RTL8019AS。其中，使用外部存储器 UT62L1024 的目的是提高单片机的数据传输速度和进行复杂的 TCP/IP 处理。同时外部的 RAM 也可以用作串行口的输入输出缓冲，以使单片机可以高速的吞吐数据。具体的电路原理框图如图 3-3 所示。

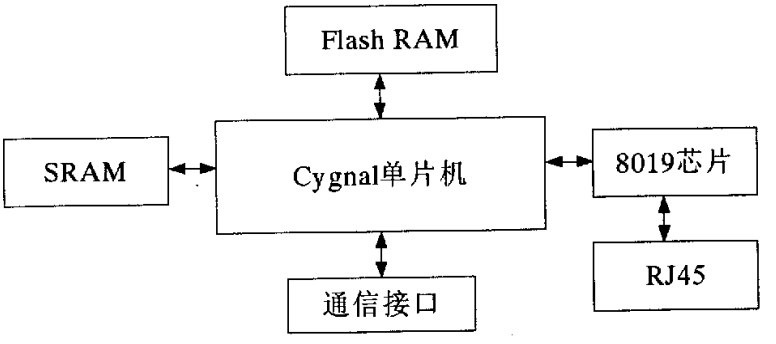

图 3-3 瘦服务器电路原理框图

3.1.2 Cygnal 单片机 C8051F020

单片机是瘦服务器中的主要功能元件之一，网卡芯片的驱动要靠单片机内程序实现。在本设计中，选用了Cygnal公司的单片机C8051F020，其主要资源如下：

（1）模拟外设

-SAR ADC

- 12位
- ±1LSB INL
- 可编程转换速率，最大100ksps
- 可多达8个外部输入；可编程为单端输入或差分输入
- 可编程放大器增益PGA：16，8，4，2，1，0.5
- 数据相关窗口中断发生器
- 内置温度传感器（±3℃）

-8位 ADC

- 可编程转换速率，最大500ksps
- 8个外部输入
- 可编程放大器增益PGA：4，2，1，0.5

-两个12位 DAC

- 可以同步输出，用于产生无抖动波形

-两个模拟比较器

-电压基准内部提供2.43V，外部基准可输入

-精确的VDD监视器和欠压检测器

（2）片内JTAG调试和边界扫描

-片内调试电路提供全速，非侵入式的在系统调试(不需仿真器)

-支持断点、单步、观察点、堆栈监视器；可以观察或修改存储器和寄存器

-比使用仿真芯片，目标仿真头和仿真插座的仿真系统有更好的性能

-符合IEEE1149.1边界扫描标准

-廉价而完全的开发套件

（3）高速8051微控制器内核

-流水线指令结构；70%的指令的执行时间为一个或两个系统时钟周期

-速度可达25MIPS（时钟频率为25MHz）

-22个矢量中断源

（4）存储器

-4352字节内部数据存储器（4k+256）

-64K字节闪速存储器可做非易失性存储；可以在系统编程,扇区大小为512

字节

-外部可扩展的64K字节数据存储器接口（可变成为复用方式或非复用方式）

（5）数字外设

-8个8位的端口I/O，所有口线均耐5V电压

-可同时使用硬件SMBus（I^2C^{TM}兼容），SPI^{TM}及两个UART串口

-可编程的16位计数器/定时器阵列PCA，有5个捕捉/比较模块

-5个通用16位计数器/定时器

-专用的看门狗定时器；双向复位引脚

（6）时钟源

-内部可编程振荡器：2-16MHz

-外部振荡器:晶体、RC、C或外部时钟

-实时钟方式（用定时器3或PCA）

（7）供电电压

-2.7～3.6V

-典型工作电流：10mA@20MHz

-多种节电休眠和停机方式

（8）100脚TQFP和64脚TQFP封装

（9）温度范围：-40℃～+85℃

C8051F020内部结构如下图3-4所示。

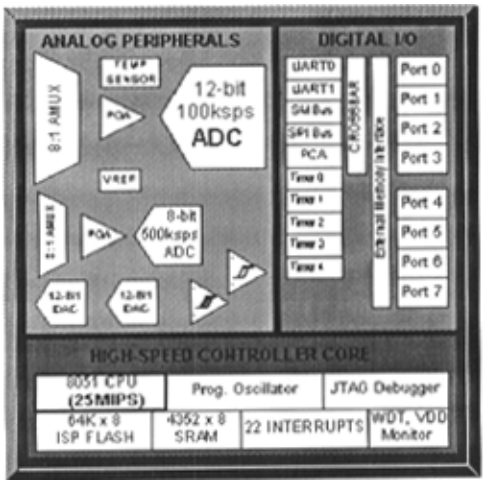

图 3-4 C8051F020 内部结构

3.1.2 以太网芯片 RTL8019AS

RTL8019AS 是 Realtek 公司生产的一种高度集成的全面支持 IEEE802.3 标准以太网控制器。只要在单片机中编写正确的驱动程序，RTL8019AS 就能实现传输网络数据的功能。

按数据链路的不同，可以将 RTL8019AS 内部划分为远程 DMA（remote DMA）通道和本地 DMA（local DMA）通道两个部分。本地 DMA 完成控制器与网线的数据交换，主处理器收发数据只需对远程 DMA 操作。当主处理器要向网上发送数据时，先赋值于 remote DMA 的起始地址寄存器 RSAR0、 RSAR1 和字节计数器 RBCR0、RBCR1，然后将一帧数据通过远程 DMA 通道送到 RTL8019AS 中的发送缓存区，发出传送命令。RTL8019AS 在完成了上一帧的发送后，再完成此帧的发送。RTL8019AS 接收到的数据通过 MAC 比较、CRC 校验后，由 FIFO 存到接收缓冲区，收满一帧后，以中断或寄存器标志的方式通知主处理器。原理框图如图 3-5 所示[30]。

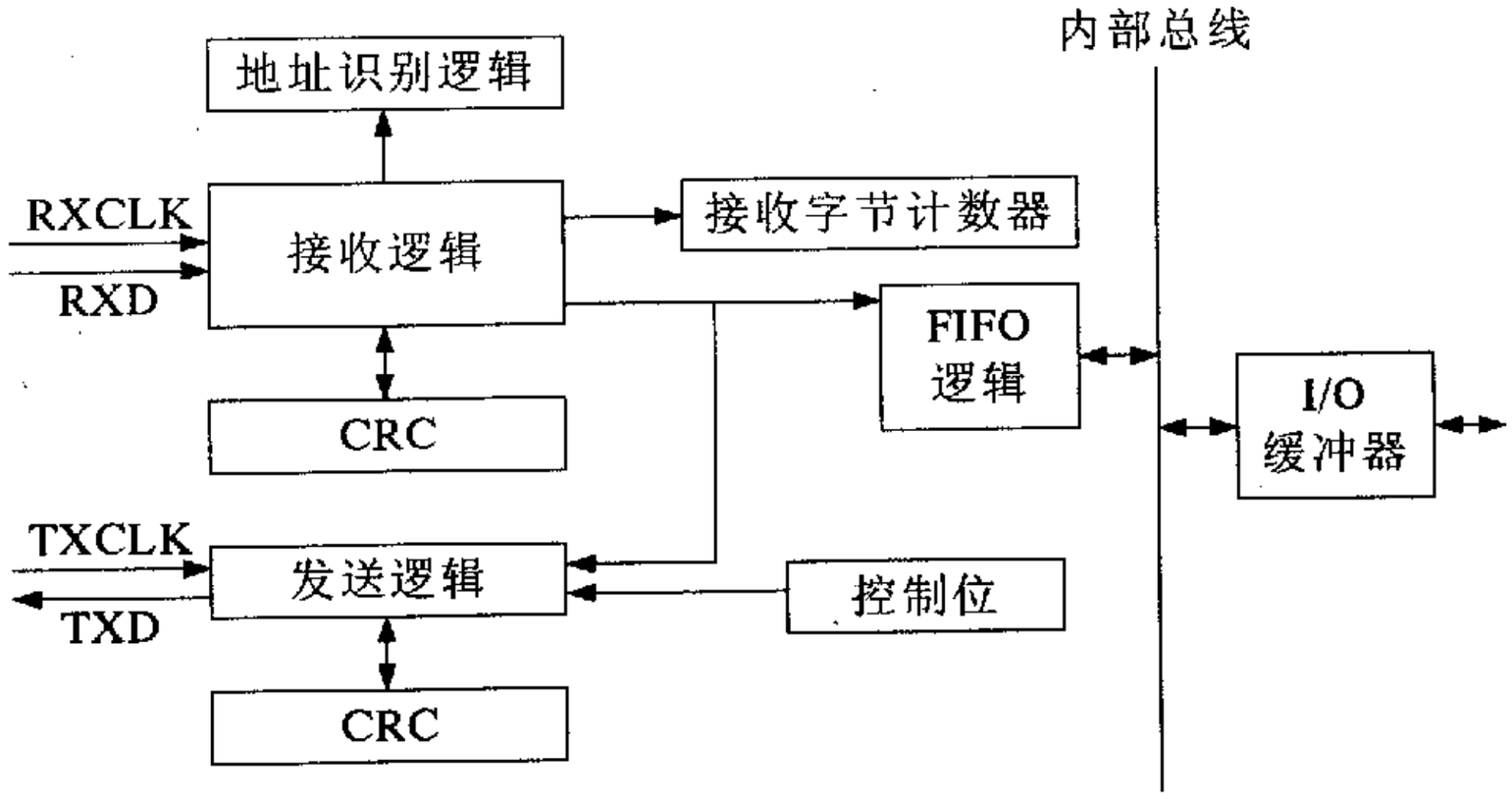

图 3-5 RTL8019AS 原理框图

在图 3-5 中，接收逻辑在接收时钟的控制下，将串行数据拼成字节送到 FIFO 和 CRC；发送逻辑将 FIFO 送来的字节在发送时钟的控制下逐步按位移出，并送到 CRC；CRC 逻辑在接收时对输入的数据进行 CRC 校验，将结果与帧尾的 CRC 比较，如不同，该帧数据将被拒收，在发送时 CRC 对帧数据产生 CRC，并附加在数据尾传送；地址识别逻辑对接收帧的目的地址与预先设置的本地物理地址进行比较，如不同且不满足广播地址的设置要求，该帧数据将被拒收；FIFO 逻

辑对收发的数据作 16 个字节的缓冲，以减少对本地 DMA 请求的频率。

RTL8019AS 内置的 16KB 的 SRAM 可划分为接收缓冲和发送缓冲两个部分。缓冲以页为单位，每页 256 个字节，16KB 的 SRAM 的页范围规定在 0X40-0X80，由 PSTART 和 PSTOP 寄存器来设定接收缓冲页的范围；由 RSAR0、1 和 RBCR0、1 寄存器来设定发送缓冲页的范围。CURR 指向接收到的帧的起始页，BNRY 指向还未读的帧的起始页。当 CURR 到达了接收缓冲页的底部，即与 PSTOP 相等时， CURR 又会自动指向到 PSTART 处[20]。与 DMA 有关的寄存器如图 3-6 所示。

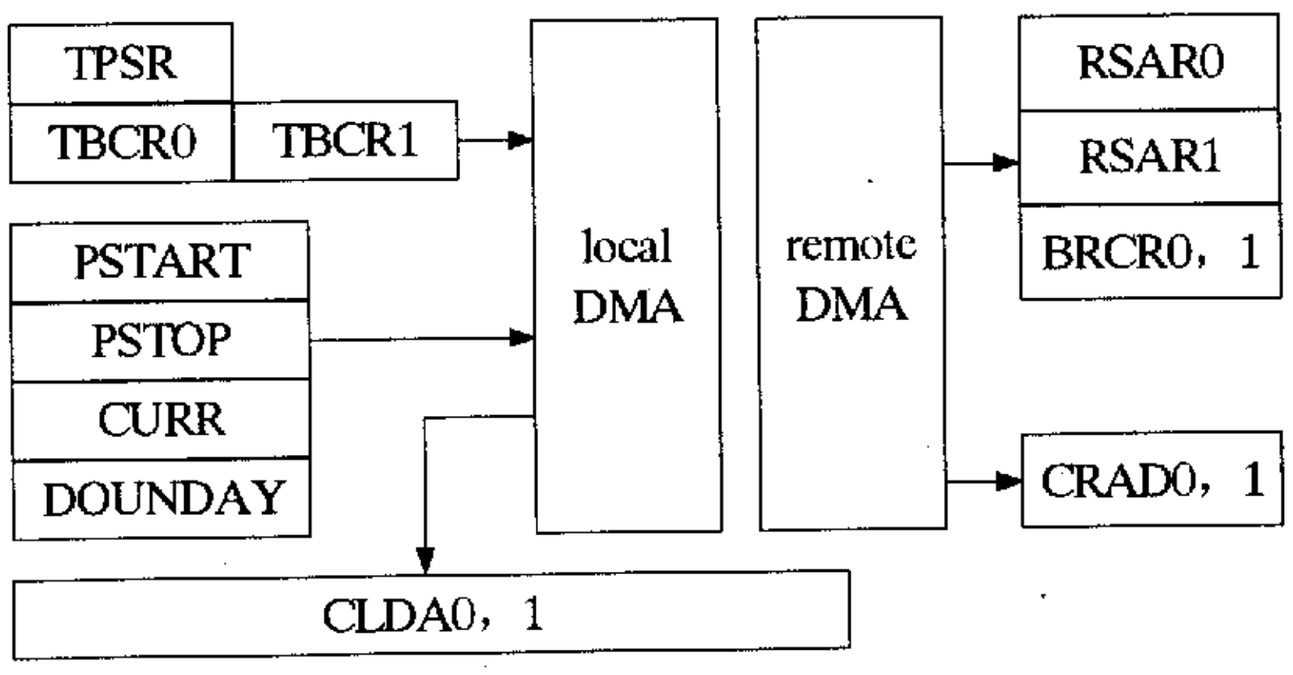

图 3-6 与 DMA 有关的寄存器

1．RTL8019AS I/O 地址空间分配

A0-A19 为 RTL8019AS 的地址线，共 20 根，我们用到的地址为十六进制的 0300H-031FH，转换成二进制为

地址线	A19-A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
300H	00000000	0	0	1	1	0	0	0	0	0	0	0	0
……	00000000	0	0	1	1	0	0	0	X	X	X	X	X
31FH	00000000	0	0	1	1	0	0	0	1	1	1	1	1

我们看到从地址 300H-31FH，地址线的高 15 位是固定不变的，A19-A5 是固定的 000000000011000，因此在编写程序时要注意只有 A4-A0 这 5 根地址线是与单片机的总线交换数据的。

另外，RTL8019AS 的输入输出地址共有 32 个，地址偏移量为 00H-1FH，分别与上述的 300H-31FH 对应。其中 00H-0FH 共 16 个地址，为寄存器地址。寄存器分为 4 页：PAGE0，PAGE1，PAGE2，PAGE3，由 8019AS 的 CR(COMMAND

REGISTER，命令寄存器)中的 PS1、PS0 位来决定要访问的页，但与 NE2000 兼容的寄存器只有前 3 页，PAGE3 是 8019AS 自己定义的，对于其他兼容 NE2000 的芯片如 DM9008 无效；10H-17H 共 8 个地址，为 DMA 地址可以用来做远程 DMA 端口（10H-17H 的 8 个地址是重复的，只要用其中的一个就可以了）；18H-1FH 共 8 个地址，为复位端口，用于给 RTL8019AS 复位。

2．RTL8019AS 内部 RAM 地址空间分配

网卡 RTL8019AS 含有 16K 字节的 RAM，地址为 0X4000-0X7FFH，是按页存储的结构，每 256 个字节称为一页，共有 64 页。页的地址就是地址的高 8 位，即页地址为 0X40-0X7FH。这 16K 的 RAM 的一部分用来存放接收的数据包，一部分用来存储待发送的数据包。

在程序中，使用 0X4C-0X7F 作为网卡的接收缓冲区，共 32 页；使用 0X40-0X4B 作为网卡的发送缓冲区，共 12 页，正好可以存储 2 个最大的以太网包。这是因为最大的以太网数据包是 1514 字节+4 字节校验，所以最大的以太网数据包需要 6 页，即 256*6=1536 字节来存储。这两个发送缓冲区的作用是：用户可以将数据包放在发送缓冲区 1 中，然后启动发送。发送的过程中，如果用户还有数据包要发，那么这时把要发的数据包放在发送缓冲区 2 中（一边发送，一边把下一包放在缓冲区里），等到发完发送缓冲区 1 的数据包就可以马上启动发送缓冲区 2 的数据包。这样可以不断地进行发送数据。

3．主要程序设计

对以太网芯片 RTL8019AS 的编程就是要对该芯片的各个寄存器进行控制，从而完成数据包的正确接收和发送。以太网的介质访问控制、CRC 校验及数据帧的接收和发送都由 RTL8019AS 自动完成，只需将 IP 包加上目的地址和源地址，再通过远程 DMA 接口对 RTL8019AS 内部 RAM 进行读写即可。

- RTL8019AS 初始化程序

初始化页 0 与页 1 的相关寄存器，页 2 的寄存器是只读的，不可以设置，页 3 的寄存器不是 NE2000 兼容的，不用设置。

（1）CR=0x21，选择页 0 的寄存器并使芯片处于停止模式进行寄存器设置。停止模式下，将不会发送和接收数据包；

（2）TPSR=0x40，发送页的起始页地址，初始化为指向第一个发送缓冲区的页即 0x40；

（3）PSTART=0x4c，PSTOP=0x80，接收缓冲区起始页和结束页地址；

（4）BNRY=0x4c，读指针，指向最后一个已经读取的页；

（5）RCR=0xe0，设置接收配置寄存器，使用接收缓冲区，仅接收自己地址的数据包（以及广播地址数据包）和多点播送地址包，小于 64 字节的包丢弃，校验错的数据包不接收；

（6）TCR=0xe2，设置发送配置寄存器，启用 CRC 自动生成和自动校验，工作在正常模式；

（7）DCR=0xc8，设置数据配置寄存器，使用 FIFO 缓存，普通模式，8 位数据 DMA；

（8）IMR=0x00，设置中断屏蔽寄存器，屏蔽所有中断；

（9）CR=0x61，选择页 1 的寄存器；

（10）CURR=0x4d，CURR 是 RTL8019AS 写内存的指针，指向当前正在写的页的下一页，初始化时指和 0x4c+1=0x4d；

（11）设置多址寄存器 MAR0～MAR7，均设置为 0x00；

（12）设置网卡地址寄存器 PAR0～PAR5；

（13）CR=0x22，选择页 1 的寄存器，进入正常工作状态。

- 数据发送程序

将待发送的数据按帧格式封装，需要设置以太网目的地址、以太网源地址、协议类型，再按所设置的协议类型来设置数据段。之后启动远程 DMA，将数据写入 RTL8019AS 的发送缓冲区，再启动本地 DMA，将数据发送网上。

程序中涉及的寄存器如下：

（1）RSRA0，RSRA1，要写入到的网卡里的 RAM 的起始地址，RSRA0 表示低 8 位，RSRA1 表示高 8 位；

（2）RBCR0，RBCR1，要连续写入的字节数，RBCR0 表示低 8 位，RBCR1 表示高 8 位；

（3）CR=0x12，选择页 0 的寄存器，进行写 DMA 操作。

- 数据接收程序

以太网的物理传输帧的接收过程分为两步：第一步由本地 DMA 将以太网传输帧存入 RTL8019AS 的接收缓冲区中；第二步由远程 DMA 将 RTL8019AS 接收缓冲区中的以太网传输帧读入主机内存。第一步由 RTL8019AS 自动完成，只需要对相关寄存器如 PSTART、PSTOP、CURR 和 BNRY 初始化，网卡就能自动接收网络上的数据包，然后写入缓冲区。第二步要通过编程实现。首先，必

须判断是否接收到了新的数据包，其次，要确定新数据包的起始地址和长度，然后，就要对 CURR 和 BNRY 这两个寄存器进行处理，最后启动 RTL8019AS 的远程 DMA 读操作。这里涉及的寄存器与数据发送程序中的一样。

3.2 网络接口层的实现

3.2.1 以太网协议简介

一个标准的以太网物理传输帧由以下部分组成，如表 4-1 所示(单位：字节)。

表 3-1 以太网数据帧格式

PR	SD	DA	SA	TYPE	DATA	PAD	FCS
7	1	6	6	2	46-1500	可选	4
同步位	分隔位	目的地址	源地址	类型字段	数据段	填充位	帧校验序列

除了数据段的长度不定外，其他部分的长度固定不变。数据段为 46～1500 字节。以太网规定整个传输包的最大长度不能超过 1514 字节（14 字节为 DA、SA、TYPE)，最小不能小于 60 字节，除去 DA、SA、TYPE14 字节，还必须传输 46 字节的数据，当数据段的数据不足 46 字节时需填充，填充字符的个数不包括在长度字段里；超过 1500 字节时，需拆成多个帧传送。事实上，发送数据时，PR、SD、FCS 及填充字段这几个数据段由以太网控制器自动产生；而接收数据时，PR、SD 被跳过，控制器一旦检测到有效的前序字段（即 PR、SD)，就认为接收数据开始。

3.2.2 程序设计概述

网络接口层是TCP/IP协议栈与下层物理设备的驱动程序之间的接口。各函数关系如图3-7所示。

发送普通数据时，本层的网络接口层发送函数eth_send( )负责接收上层协议产生的数据，并调用send_frame( )函数实现发送。

接收数据时，eth_rcve( )首先判断接收数据类型，如果是IP数据报，则调用上层IP协议中的ip_recv( )函数；如果是ARP数据报，则调用ARP报文处理函数arp_recv( )。

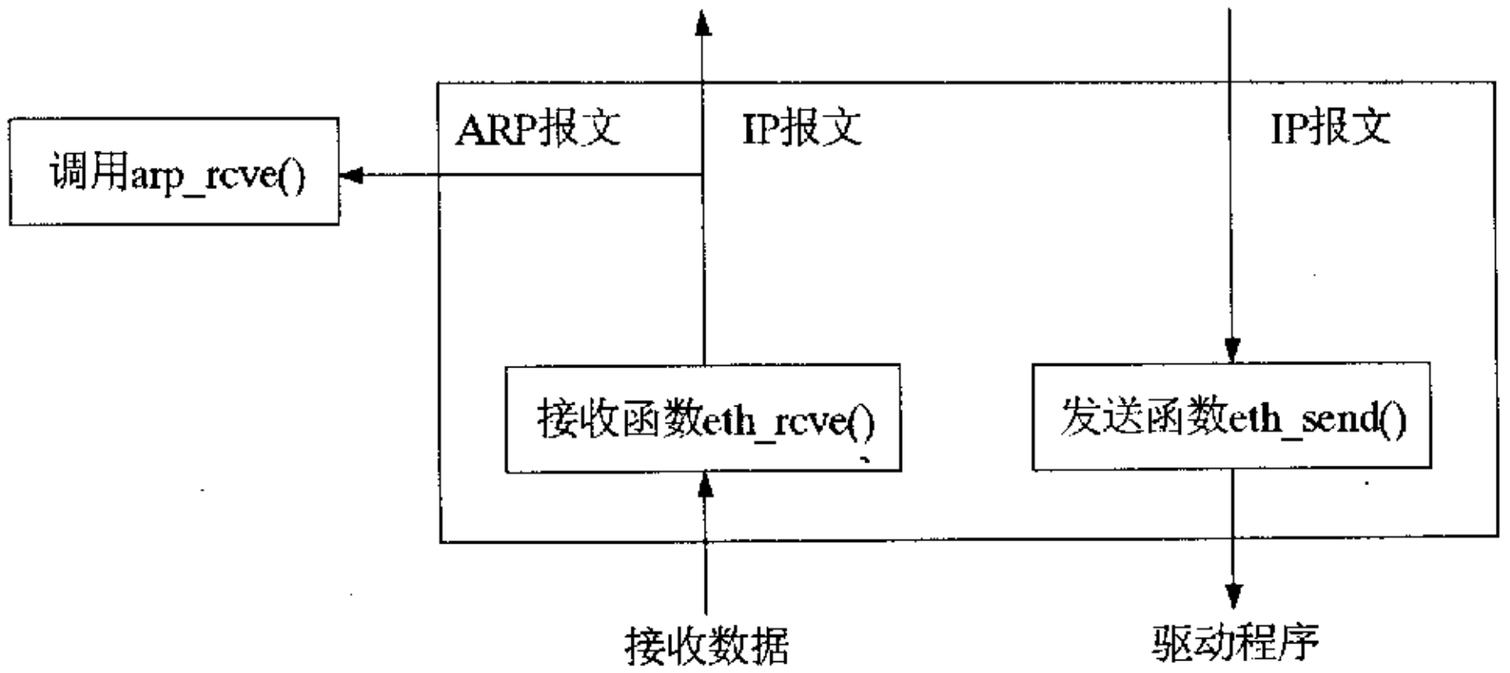

图 3-7 网络接口层函数关系

3.2.3 主要数据结构

```
typedef struct
{
    UCHAR dest_hwaddr[6]; /*目的硬件地址*/
    UCHAR source_hwaddr[6];   /*源硬件地址*/
    UINT   frame_type;  /*帧类型*/
} ETH_HEADER;
```

3.2.4 函数详细设计

1. eth_send( )：发送数据

直接调用send_frame( )函数发送一个数据。

2. eth_rcve( )：接收数据

处理接收到的以太网帧，判断帧类型，如果是ARP数据报就调用arp_rcve( )函数；如果是IP数据报就调用ip_rcve( )函数。

3.3 ARP 协议详细设计

3.3.1 ARP 协议简介

地址解析协议（Address Resolution Protocol ，ARP）主要是进行逻辑地址和物理地址之间的动态映射。以太网的网络接口支持一种独特的 48 位以太网地址。然而，IP 网却使用 32 位的逻辑地址。IP 网采用一种层次寻址模式，单个的网络

节点是一个子网的一部分，反过来，子网又是一个更大子网的一部分。以太网则利用一种独立于逻辑寻址的寻址模式——物理地址，以太网帧的传送是由唯一的以太网物理地址来指定的。因此地址解析协议（ARP）就成为了从 IP 地址映射到物理地址的桥梁。它提供了一种机制，使得一个网络节点能够在以太网上给 IP 地址已知的目的节点传送 IP 包。地址解析协议（ARP）必须和网络接口层直接接口，因为所有其它的网络层都用 IP 地址而不是物理地址。

3.3.2 程序设计概述

ARP协议实现对ARP报文的解释、处理，同时管理ARP映射表。

ARP报文分为ARP请求报文和ARP应答报文。ARP报文接收函数arp_rcve( )负责对输入ARP报文进行处理，如果收到ARP请求报文，则调用函数arp_send( )发出ARP应答报文；如果收到ARP应答报文，且请求解析的物理地址正是自己的物理地址，则调用函数ip_send( )把等候地址解析的IP数据报发出。

管理ARP映射表的工作主要包括：ARP表的查找，ARP表项的添加、删除以及ARP 表项的更新。其中，有一个定时器负责对ARP表项的更新工作。每当收到一个ARP报文后，首先确定目的IP地址与本地IP地址相符，然后以该IP地址为索引查找ARP映射表，如果找到就重启定时器；如果没找到且ARP映射表还有空的表项，则首先以本次IP地址-硬件地址对更新ARP映射表中的表项，再启动定时器，否则就更新表项中最陈旧的一项。ARP协议的函数关系如图3-8所示。

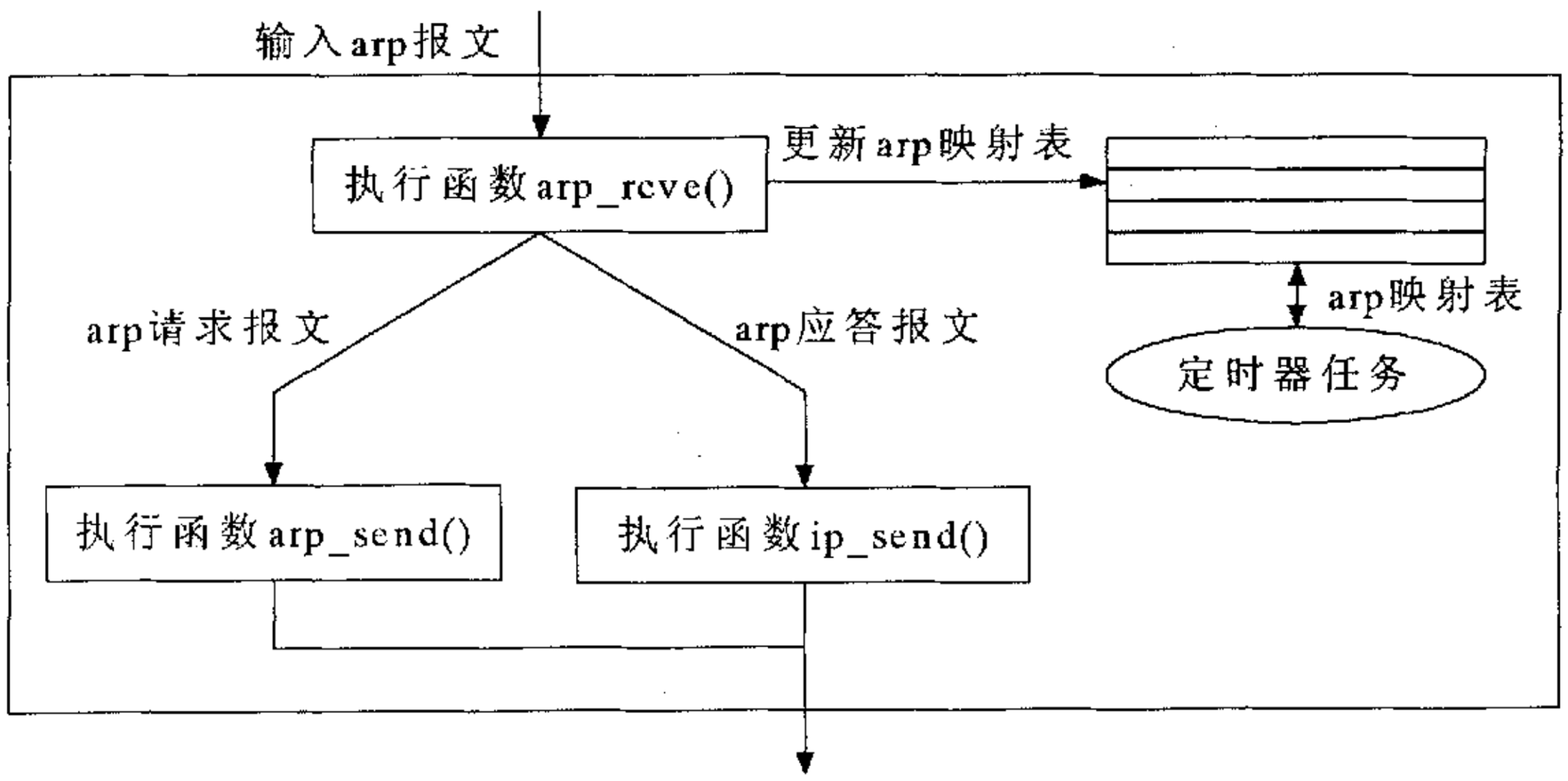

图 3-8 ARP 协议的函数关系

3.3.3 主要数据结构

1．ARP 映射表项数据结构

```
typedef struct
{
    ULONG ipaddr; /*IP地址*/
    UCHAR hwaddr[6];/*物理地址*/
    UCHAR timer;  /*生存时间*/
} ARP_CACHE;
```

2．ARP数据报首部结构

```
typedef struct
{
    UINT   hardware_type;   /*硬件类型*/
    UINT   protocol_type;   /*协议类型*/
    UCHAR hwaddr_len;/*硬件地址长度*/
    UCHAR ipaddr_len; /*协议地址长度*/
    UINT   message_type;   /*ARP报文类型*/
    UCHAR source_hwaddr[6];   /*源硬件地址*/
    ULONG source_ipaddr;        /*源协议地址*/
    UCHAR dest_hwaddr[6]; /*目的硬件地址*/
    ULONG dest_ipaddr;/*目的协议地址*/
} ARP_HEADER;
```

3.3.4 函数详细设计

1．arp_send（）：发送ARP数据包

给发送的完整数据报文分配内存，包括以太网首部和ARP首部共42字节，其中前14字节为以太网首部，接着是28字节的ARP请求/应答报文。如果是一个ARP请求报文，则是一个广播报文，如果是ARP应答报文，则该报文要送往一个指定的目的硬件地址。调用eth_send( )函数发送报文。流程图如3-9所示。

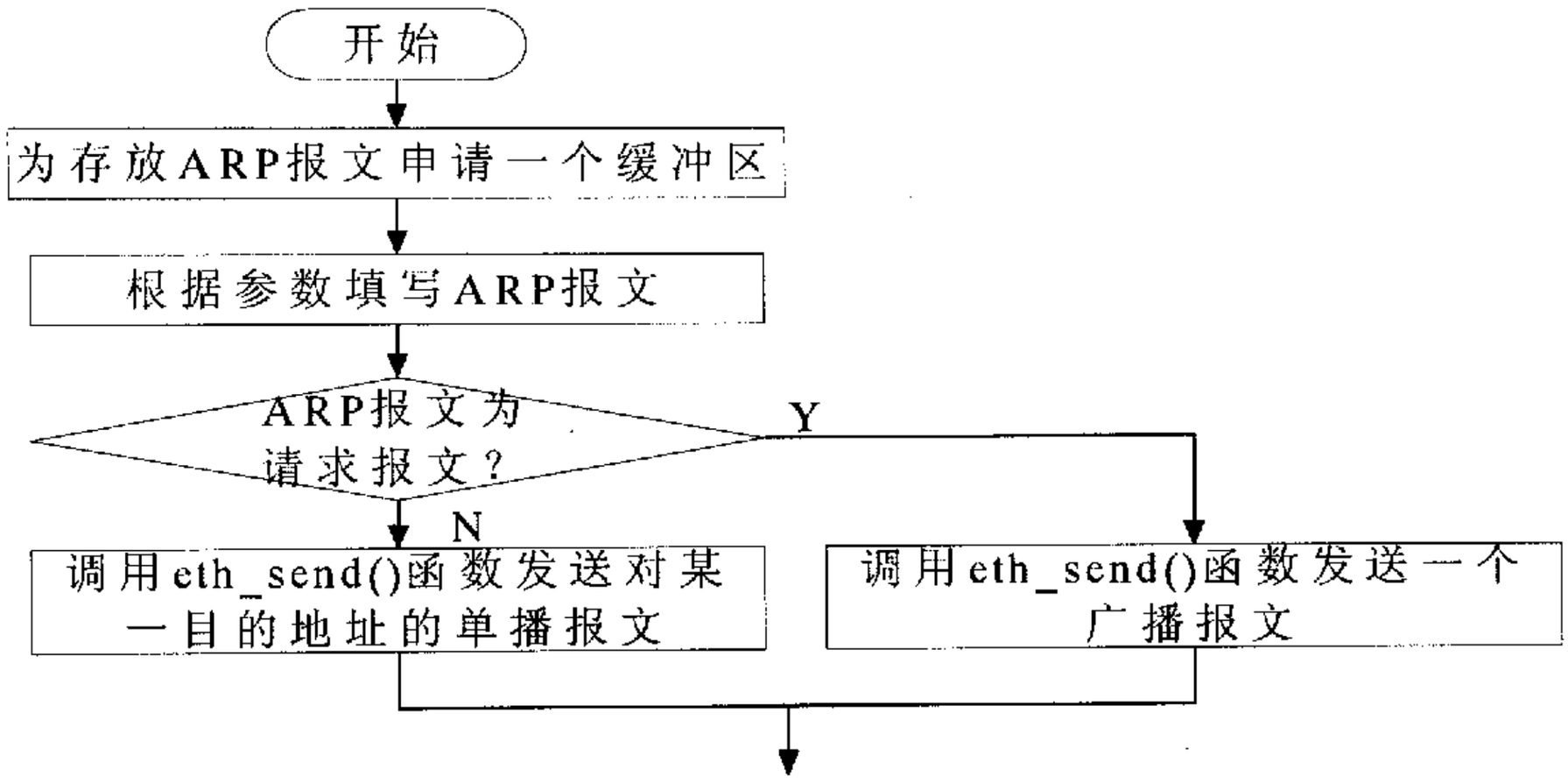

图 3-9 arp_send（）函数流程图

2．arp_rcve（）：接收ARP数据包

用来处理接收报文。首先确定收到的报文的目的IP与本地IP相符，然后以该IP地址为索引查找ARP映射表，如果找到就重启定时器；如果没找到且ARP映射表还有空项，则首先以该报文中的有用信息更新ARP映射表中的表项，再启动定时器，否则就更新表项中最陈旧的一项。如果收到ARP请求报文，则调用函数arp_send( )发出ARP应答报文；如果收到ARP应答报文，且请求解析的物理地址正是自己的物理地址，则调用函数ip_send( )把等候地址解析的IP数据报发出。流程图如3-10所示。

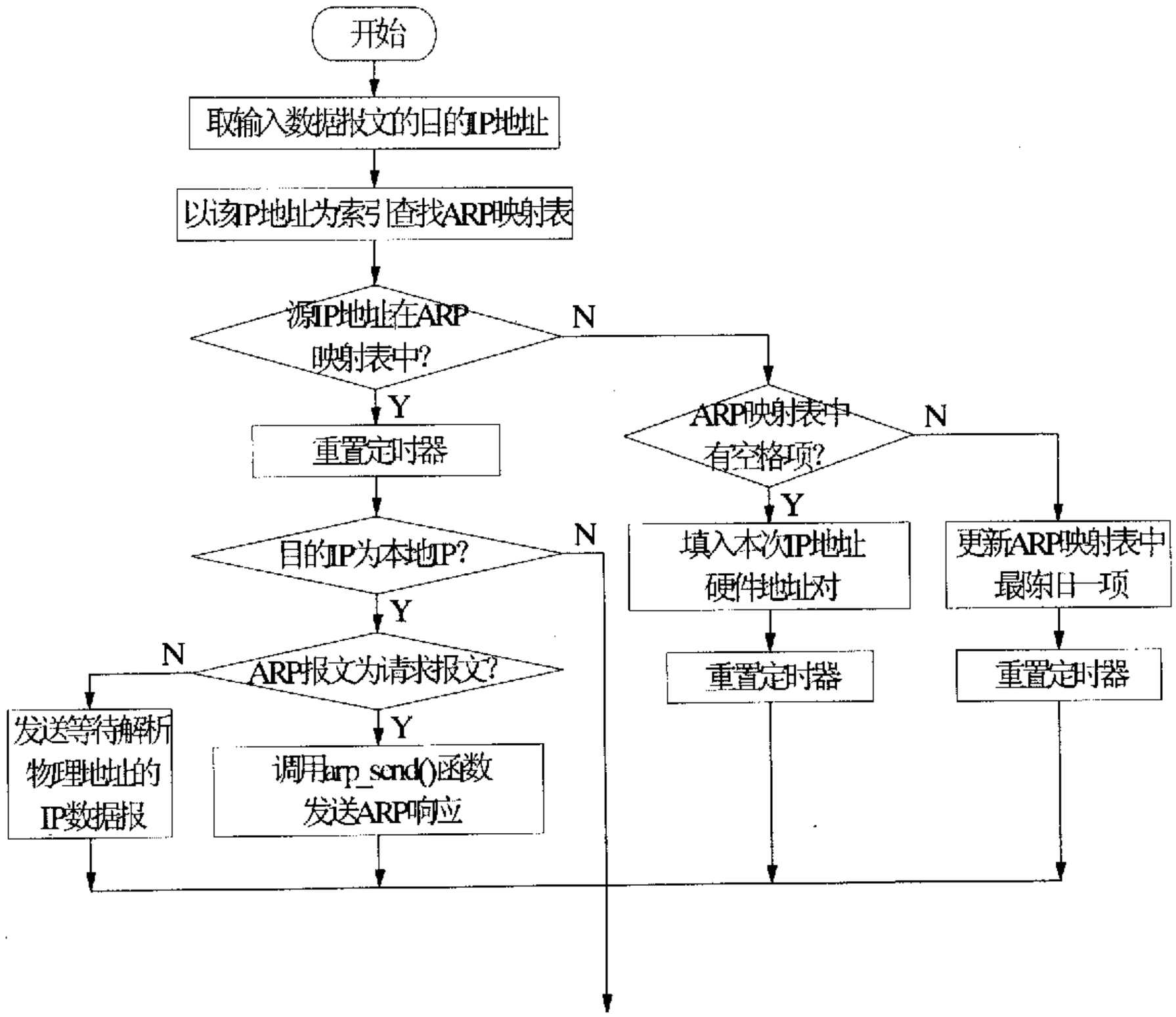

图 3-10 arp_rcve（）函数流程图

3．arp_retransmit（）：重发ARP数据包

如果对第一次ARP请求没有响应就重发一个ARP请求。该函数每0.5秒被调用一次。如果两次重发后都没有响应，则要传送的数据包将丢弃。

4．age_arp_cache（）：ARP 高速缓存超时函数

每60秒调用一次来更新ARP映射表，如果表项超时将被删除。

5．arp_resolve（）：ARP 地址解析

本函数的主要工作是找到所给的IP地址的以太网硬件地址。如果目的IP地址在子网内，就找到目的以太网硬件地址，否则找到网关的以太网地址。返回指针指向硬件地址或NULL（如果没找到）。

3.4 IP 协议详细设计

3.4.1 IP 协议简介

网际协议（Internet Protocol ， IP）是TCP/IP协议中最为核心的协议。所有的TCP，UDP，ICMP及IGMP数据都以IP数据报格式传输。IP提供不可靠的，无连接的数据报传输。不可靠（unreliable）是指它不能保证IP数据报能成功地到达目的地，IP仅提供尽可能好的传输服务。无连接（connectionless）是指IP并不维护任何关于后续数据报的状态信息。每个数据报的处理是相互独立的，IP数据报可以不按发送顺序接收。

1．IP地址与物理地址

主机的IP地址与物理地址是不同的，在局域网中物理地址已经固化在网卡的ROM中（其他的不一定）。IP地址放在IP数据报的首部，而物理地址则放在MAC帧的首部。这两种地址的使用是有区别的：IP地址只在网络层以上使用，物理地址只在数据链路层以下使用[25]。其间的区别参见图3-11。

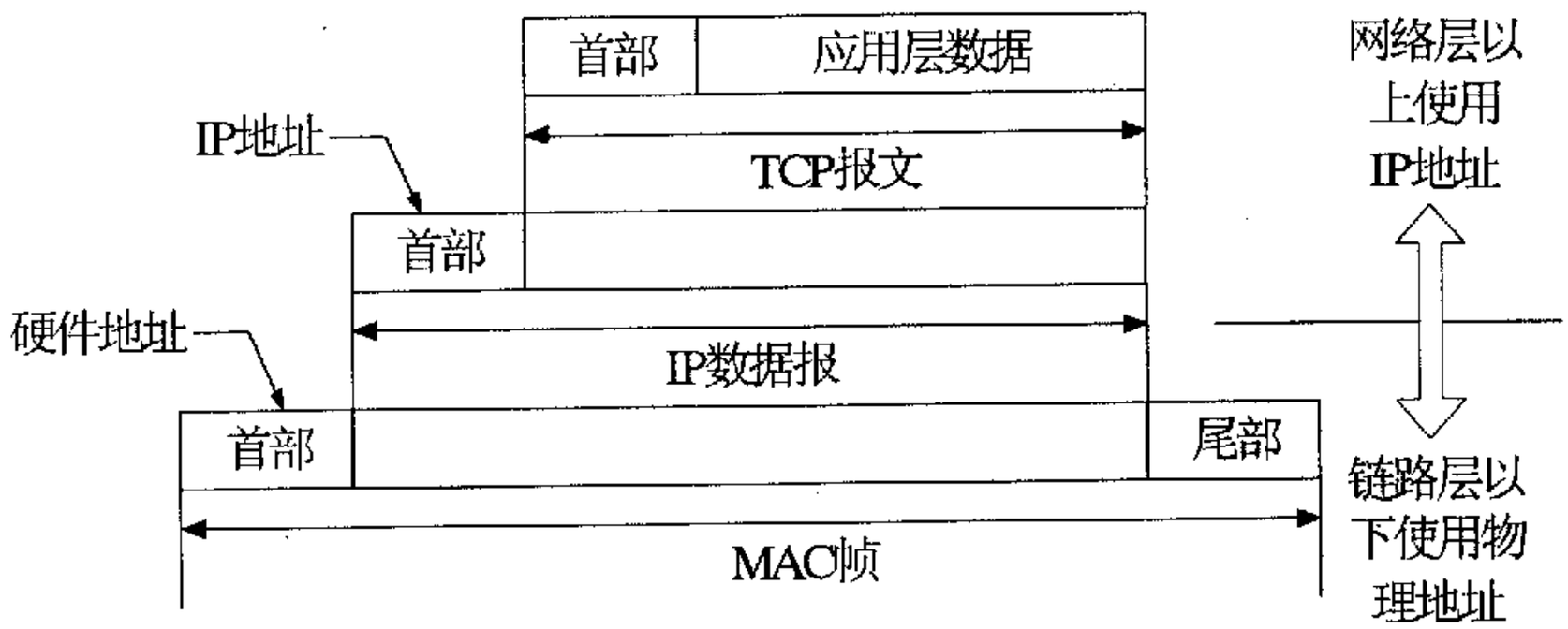

图 3-11 IP 地址与物理地址的区别

2．IP路由选择

从概念上说，IP路由选择是简单的，特别对于主机来说。如果目的主机与源主机直接相连（如点对点链路）或都在一个共享网络上（如以太网或令牌环网），那么IP数据报就可以直接送到目的主机上。否则，主机把数据发往一默认的路由器上，由路由器来转发该数据报。在一般的体制中，IP可以从TCP，UDP，ICMP和IGMP接收数据报（即在本地生成的数据报）并进行发送，或者从一个接口接收数据报（即待转发的数据报）并进行发送。IP层在内存中有一个路由表。当收

到一份数据报并进行发送时，它要对该表进行一次搜索。当数据报来自某个网络接口时，IP首先检查目的IP地址是否为本机的IP地址之一或IP广播地址。如果确实是这样，数据报就被送到由IP首部协议字段所指定的协议模块进行处理。如果数据报的目的不是这些地址，那么如果IP层被设置为路由器功能，那么就对数据报进行转发；否则，数据报就被丢弃。IP路由选择是逐跳地（hop-by-hop）进行的。IP路由表主要包含以下信息：①目的IP地址；②下一站（或下一跳）路由器的IP地址；③标志；④为数据报的传输指定一个网络接口。由路由表包含的信息可以看出，IP并不知道到达任何目的的完整路径。所有的IP路由选择只为数据报传输提供下一站路由器的IP地址。IP路由选择主要完成以下功能：①搜索路由表，寻找与目的IP地址完全匹配的表目；②搜索路由表，寻找能与目的网络号匹配的表目；③搜索路由表，寻找标为“默认”的表目[26]。

3.4.2 程序设计概述

在本设计中，IP协议实现对无连接的IP数据报的寻径、传送。

发送数据时，上层的UDP发送函数udp_send( )以及同层的ICMP协议中的ping_send( )函数把所发数据交给ip_send( )函数。在ip_send( )函数中，调用arp_resolve( )函数，查找ARP映射表获得对应的物理地址，如果找到相应的路由，则调用网络接口层发送函数eth_send( )实现发送；如果没有找到相应的路由，则发送ARP请求报文，并设置wait结构体。

接收数据时，函数ip_rcve( )被下层的网络接口层接收函数eth_rcve( )调用以接收IP数据报。在ip_rcve( )中，检查数据报的目的IP地址，如果不是自己的IP地址，则程序返回；如果为自己的IP地址，则再检查该数据报的类型，如果是UDP报文，则调用上层的UDP数据接收函数udp_rcve( )；如果是ICMP报文，则调用ICMP协议实现中的接收处理函数icmp_rcve( )。

IP协议的函数关系如图3-12所示。

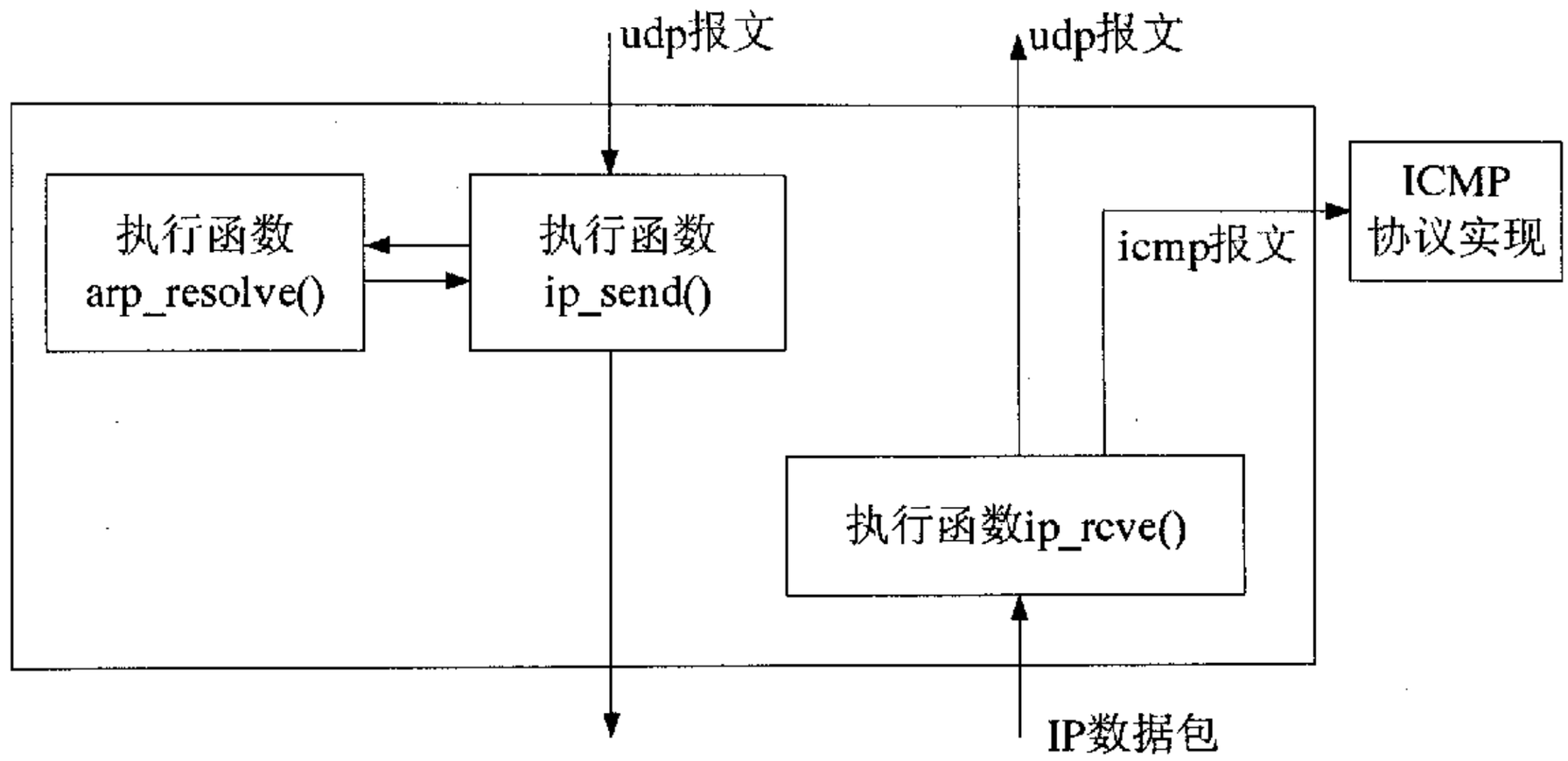

图 3-12 IP 协议的函数关系

协议标准规定，任何实现IP的程序都必须能够将数据报分片及重组。事实上，由于嵌入式系统生成的数据报足够小，能够直接通过网络传输，因此，嵌入式系统并不需要将输出数据报进行分片。

3.4.3 主要数据结构

1．wait结构体数据结构

```
typedef struct
{
    UCHAR xdata * buf;
    ULONG ipaddr;
    UCHAR proto_id;
    UINT   len;
    UCHAR timer;
} WAIT;
```

2．IP数据报首部结构

```
typedef struct
{
    UCHAR ver_len;      /*协议版本号及首部长度*/
    UCHAR type_of_service;/*服务类型*/
```

```
    UINT    total_length; /*数据报总长度*/
    UINT    identifier;    /*标识字段*/
    UINT    fragment_info;   /*片偏移*/
    UCHAR time_to_live;    /*生存时间*/
    UCHAR protocol_id;/*协议字段*/
    UINT    header_cksum;   /*首部校验和*/
    ULONG source_ipaddr;   /*源IP地址*/
    ULONG dest_ipaddr;/*目的IP地址*/
} IP_HEADER;
```

3.4.4 函数详细设计

1．ip_send()：发送IP数据

负责处理输出IP数据报。加20字节的IP首部和校验和，并将IP数据包发送到网络接口层。以输入的源、目的地址封装报文，并计算头标校验和，形成IP报文。然后调用网络接口层的发送函数eth_send()完成发送。流程图如3-13所示。

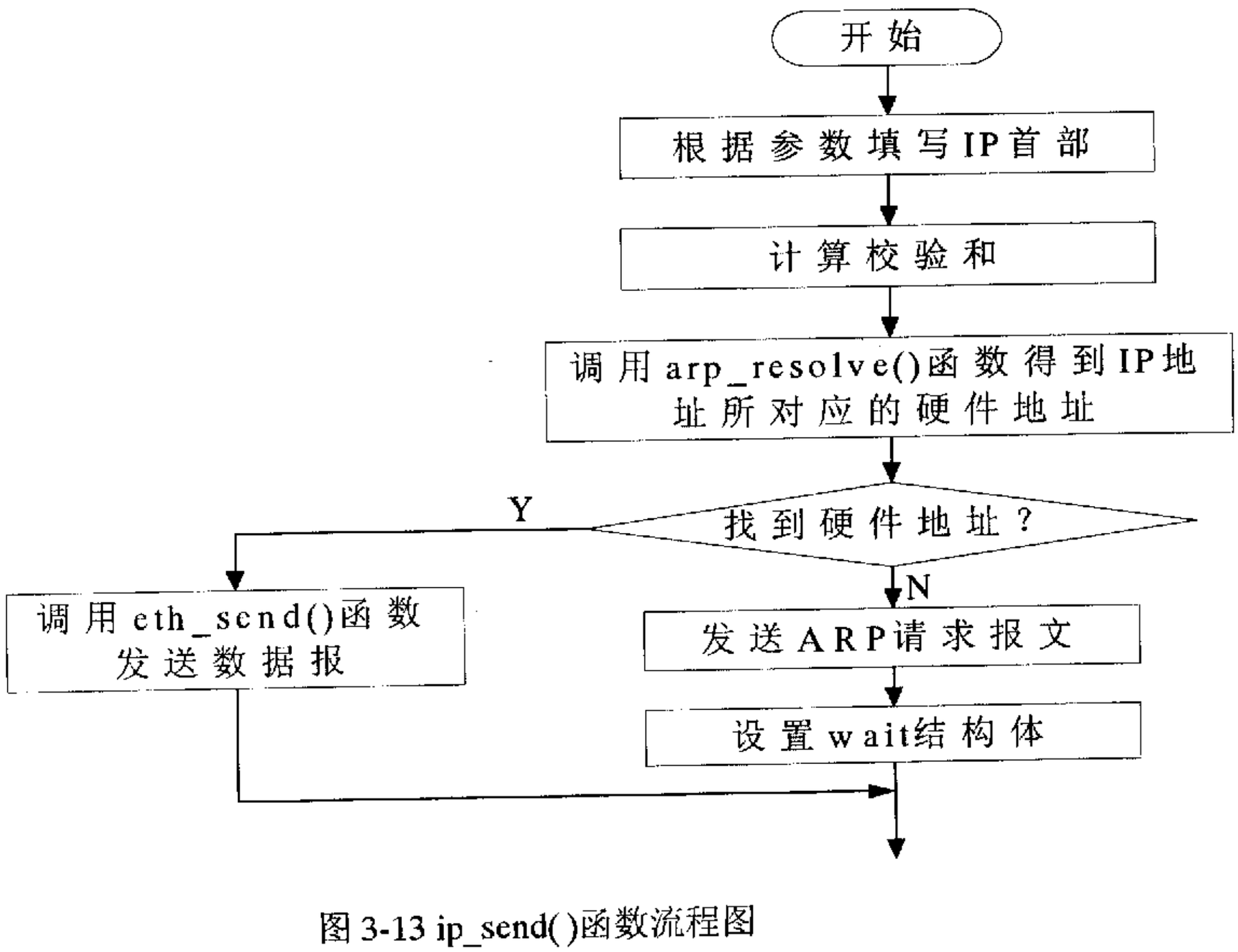

图 3-13 ip_send()函数流程图

具体函数代码如下：

```
void ip_send(UCHAR xdata * outbuf, ULONG ipaddr, UCHAR proto_id, UINT
len)
{   IP_HEADER xdata * ip;
    UCHAR xdata * hwaddr;
    static UINT ip_ident = 0;

    ip = (IP_HEADER xdata *)(outbuf + 14);
    ip->ver_len = 0x45;                  // IPv4版本号为4，字段值为5
    ip->type_of_service = 0;
    ip->total_length = 20 + len;
    ip->identifier = ip_ident++;         //每发送一份报文标识字段的值加1
    ip->fragment_info = 0;                // 数据报不分片
    ip->time_to_live = 32;                // TTL（生存时间）
    ip->protocol_id = proto_id;
    ip->header_cksum = 0;
    ip->source_ipaddr = my_ipaddr;
    ip->dest_ipaddr = ipaddr;
        ip->header_cksum = ~cksum(outbuf + 14, 20); /*计算并插入校验和到IP首
部，IP首部长度为20字节*/
        hwaddr = arp_resolve(ip->dest_ipaddr);/*通过ARP协议查找到目的硬件地
址*/
        // Null表示没找到硬件地址，则发送一个ARP请求
        if (hwaddr == NULL)
        {
            /*填充WAIT结构体，当ARP响应到达时能根据它判断我们收到的报
文形式*/
            wait.buf = outbuf;
            wait.ipaddr = ip->dest_ipaddr;
            wait.proto_id = proto_id;
            wait.len = len;
```

```
            wait.timer = ARP_TIMEOUT;
            return;
        }
        eth_send(outbuf, hwaddr, IP_PACKET, 20 + len);
    }
```

2．ip_rcve( )：接收IP数据

负责处理来自于链路层的IP数据报。首先确定数据报目的地址为本地地址，校验和正确，协议版本号为4，数据无分片（因为本系统程序不支持IP数据报分片处理），再根据IP报文中协议字段调用其他程序进行数据报的处理。流程图如3-14所示。

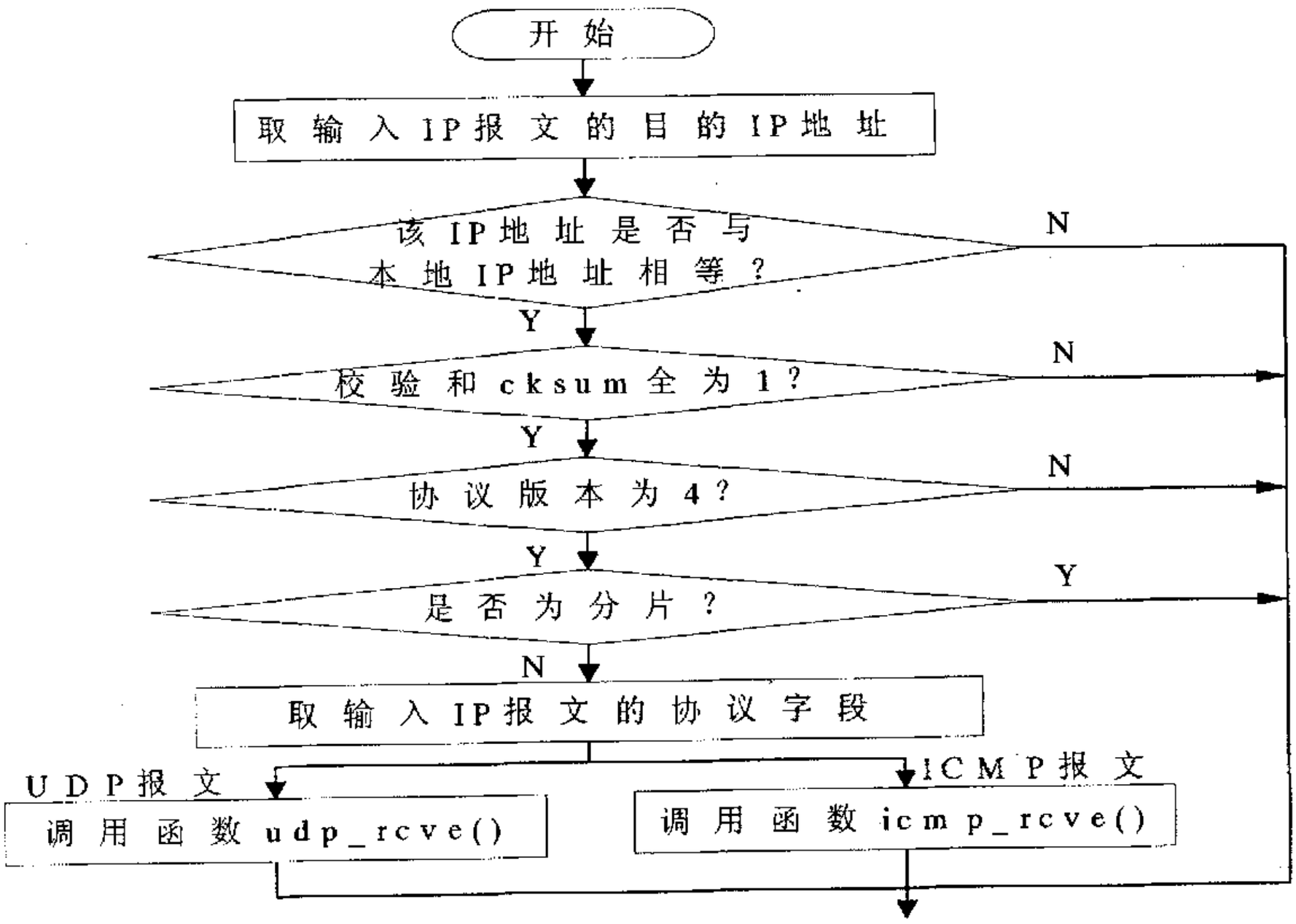

图 3-14 ip_rcve( )函数流程图

具体函数代码如下：

```
void ip_rcve(UCHAR xdata * inbuf)
{
    IP_HEADER xdata * ip;
    UINT idata header_len, payload_len;
```

```
ip = (IP_HEADER xdata *)(inbuf + 14);

if (ip->dest_ipaddr != my_ipaddr) return;//确定IP地址为本地地址

header_len = 4 * (0x0F & ip->ver_len);
payload_len = ip->total_length - header_len;
if (cksum(inbuf + 14, header_len) != 0xFFFF)
{
    if (debug) serial_send("IP:    Error, cksum bad\r");
    return;
}//验证校验和

if ((ip->ver_len >> 4) != 0x04)
{
    if (debug) serial_send("IP:    Error, not IPv4\r");
    return;
}//确定IP协议版本号为4

if ((ip->fragment_info & 0x3FFF) != 0)
{
    if (debug) serial_send("IP:    Error, fragmented msg rcvd\r");
    return;
}//确定数据报报文不分片，因为上层采用UDP协议，为防止出错不分片

memmove(inbuf + 34, inbuf + 14 + header_len, payload_len);//防止信息被覆盖
header_len = 20;
ip->ver_len = 0x45;
ip->total_length = 20 + payload_len;
}

switch (ip->protocol_id)
{
    case ICMP_TYPE:
        icmp_rcve(inbuf, payload_len)
        break;
    case UDP_TYPE:
        udp_rcve(inbuf, payload_len);
        break;
    }//查看协议ID号，调用相应的函数处理数据报报文
}
```

3.5 ICMP 协议详细设计

3.5.1 ICMP 协议简介

与网际协议IP一起工作的一个协议是网际控制消息协议(Internet Control Message Protocol，ICMP)，它也位于IP层，负责接收、解释、发送ICMP报文。。在把消息从发送端传输到接收端的过程中可能会出现许多问题，例如生存时间TTL定时器到时，网关设备把数据包错送到其他地方等问题。让发送者知道数据包传输过程中出现的问题是很重要的事情，ICMP协议就是为这个目的开发的协议。

IP 并非设计为绝对可靠，ICMP 这个协议的目的是为了当网络出现问题的时候返回控制信息，而不是使 IP 协议变得绝对可靠，它并不保证数据报或控制信息能够返回，一些数据报仍将在没有任何报告的情况下丢失。上层协议必须使用自己的差错控制程序来判断通信是否正确。

3.5.2 程序设计概述

ICMP报文主要有两种功能：报告出错信息、传送控制信息。尽管与IP协议处于同一层次，但ICMP报文却是封装在IP数据报的数据段部分进行传送的，具体如图3-15所示。

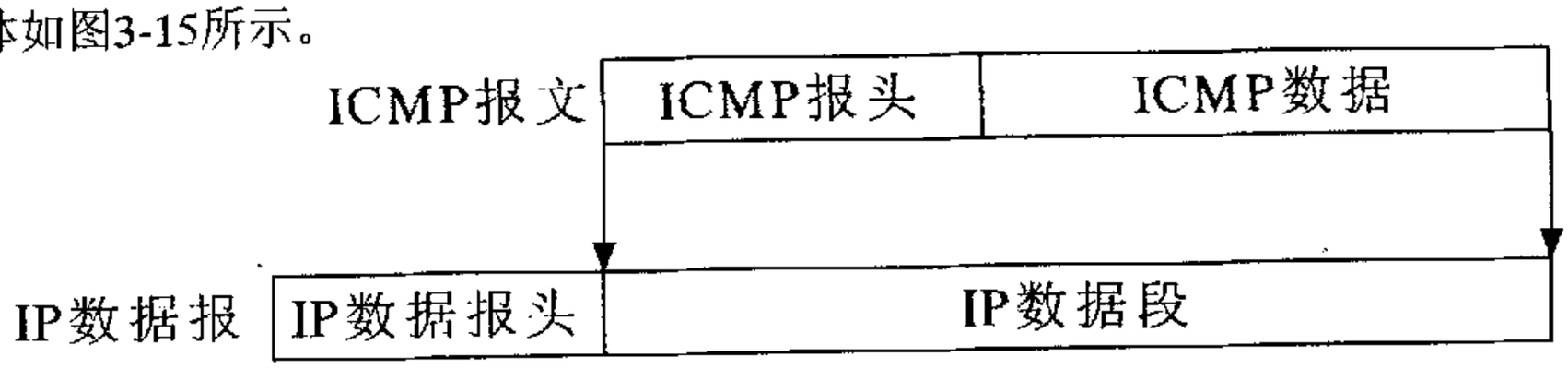

图 3-15 ICMP 报文的封装

ICMP协议实现比较单一，由三个函数组成：icmp_rcve()函数、ping_send()函数、dest_unreach_send()函数。其中，icmp_rcve()函数负责解释、处理收到的ICMP报文；ping_send()函数负责构成ICMP报文并发送该报文；dest_unreach_send()函数构成端口不可达报文并发送该报文。

3.5.3 主要数据结构

1．ICMP地址掩码请求与应答数据报文结构：

```
typedef struct
{
    UCHAR msg_type;       /*报文类型*/
    UCHAR msg_code;  /*报文代码*/
    UINT   checksum;   /*校验和*/
    UINT   identifier;    /*标识符*/
    UINT   sequence;    /*序列号*/
    UCHAR echo_data;
} PING_HEADER;
```

2．ICMP端口不可达差错报文结构

```
typedef struct
{
    UCHAR msg_type;  /*报文类型*/
    UCHAR msg_code;  /*报文代码*/
    UINT   checksum;   /*校验和*/
    ULONG msg_data;
    UCHAR echo_data;
} ICMP_ERR_HEADER;
```

3.5.4 函数详细设计

1．dest_unreach_send( )：目的主机无法达到处理函数

负责建立一个输出ICMP目的端口不可达应答报文。如果收到一个UDP数据报而且目的端口与某个正在使用的进程不相符。那么UDP就返回一个ICMP端口不可达报文。流程图如3-16所示。

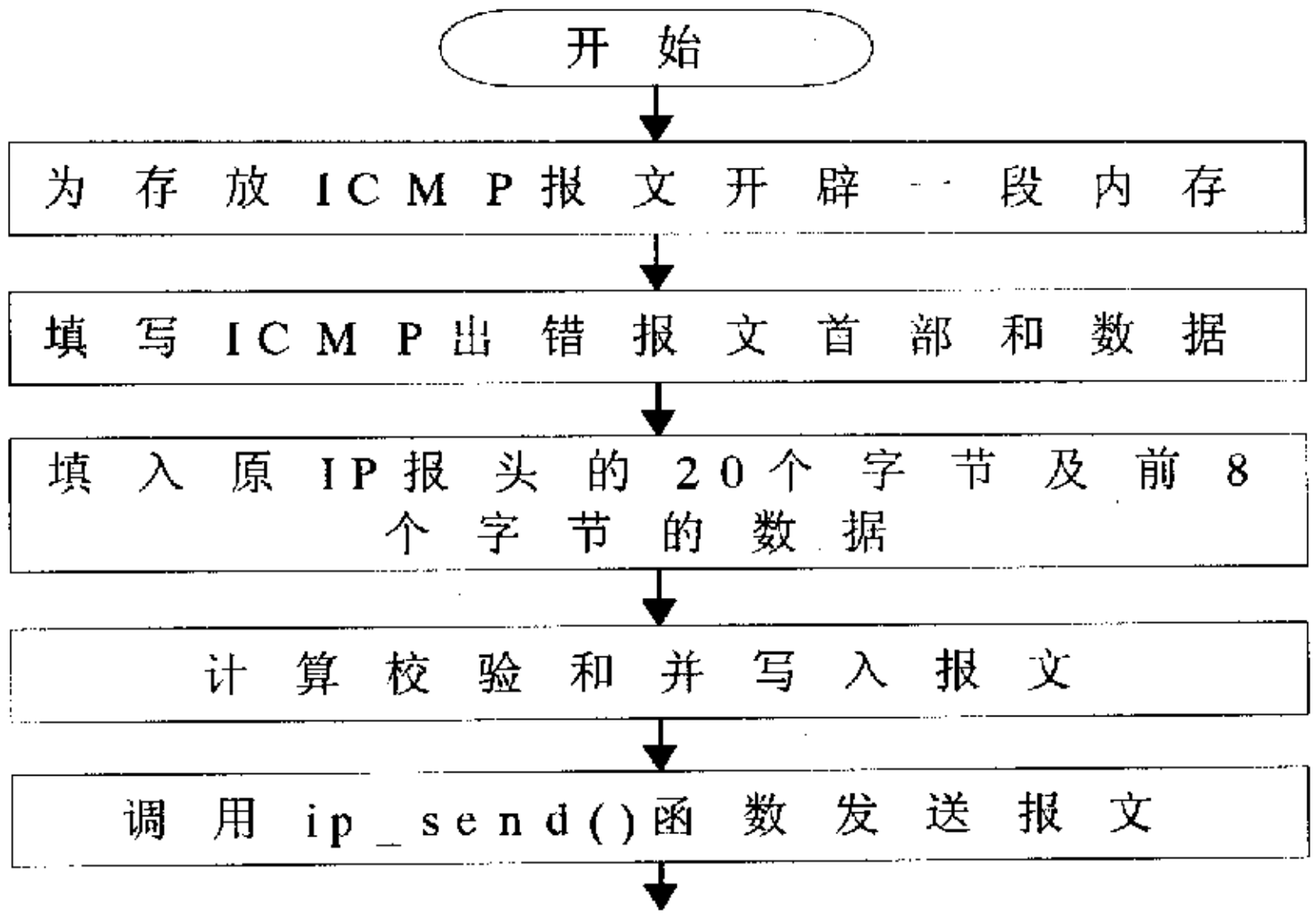

图 3-16 dest_unreach_send( )函数流程图

2．ping_send( )：发送ICMP回显请求给目的主机，并等待返回ICMP回显应答

负责建立一个PING应答报文。本函数开辟了一段内存用来存放完整的输出报文，包括以太网首部和IP首部。流程图如3-17所示。

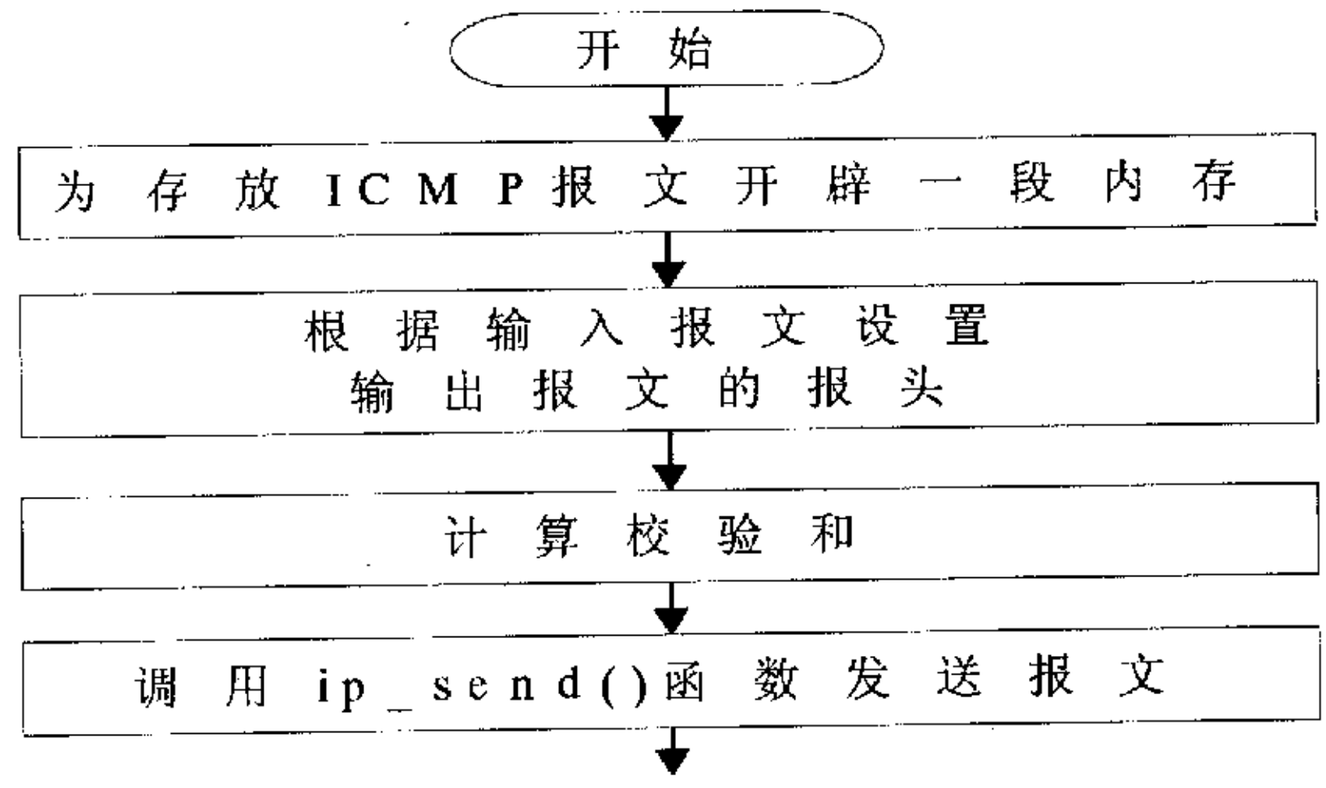

图 3-17 ping_send( )函数流程图

3．icmp_rcve( )：接收ICMP回显数据

负责处理输入的ICMP报文。首先确认校验和正确，然后再根据报文类型分别处理。流程图如3-18所示。

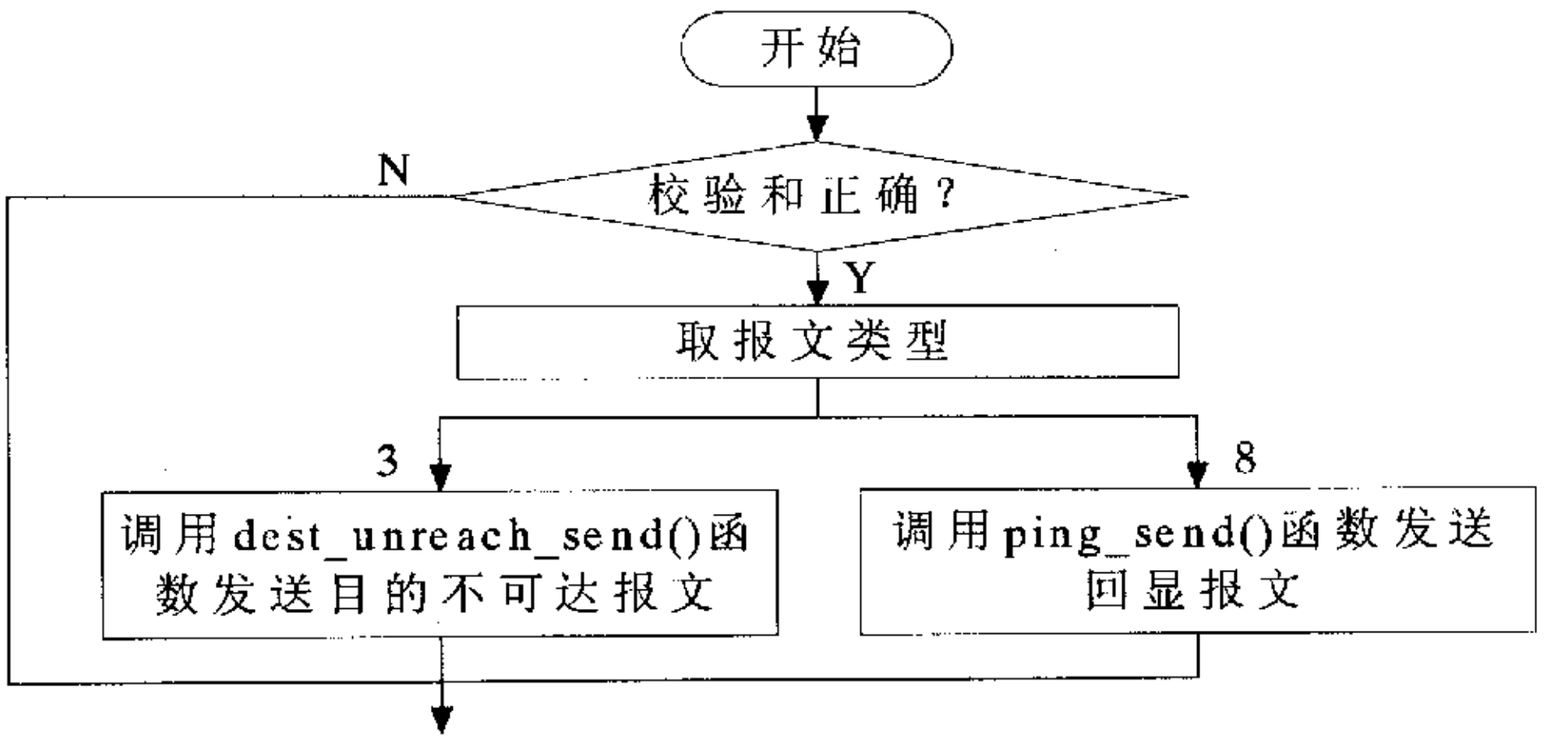

图 3-18 icmp_rcve( )函数流程图

3.6 UDP 协议详细设计

3.6.1 UDP 协议简介

用户数据报协议（User Datagram Protocol ， UDP）是一个简单的面向数据报的传输层协议，提供应用程序之间的无连接通信：进程的每个输出都正好产生一个 UDP 数据报，并组装成一份待发送的 IP 数据报。UDP 不提供可靠性，它把应用程序传给 IP 层的数据发送出去，但是并不保证它们能到达目的地[27]。

3.6.2 程序设计概述

许多基于UDP的应用程序在高可靠性、低延迟的局域网上运行得很好，而一旦到了通讯子网QoS很低的网间网环境下，可能根本不能运行。但是正是由于UDP没有保证可靠性的机制，没有其他的关卡机制，UDP才得以实现全速地发送（即充分发挥物理通信设备的速度）。所以，UDP适合于微处理器能力有限、实时要求较高且可靠性要求不高的嵌入式应用[37]。而本文设计的瘦服务器工作在印刷机套色控制系统中，它是一个高可靠性、低延迟的局域网，并且网络传送的数据量不大，数据包不分片，所以只需通过计算校验和来保证数据的正确性。

1．UDP报文及其封装

一条UDP报文就叫一条用户数据报，在概念上分为两部分：头标和数据区。其格式如图3-19所示[2]。

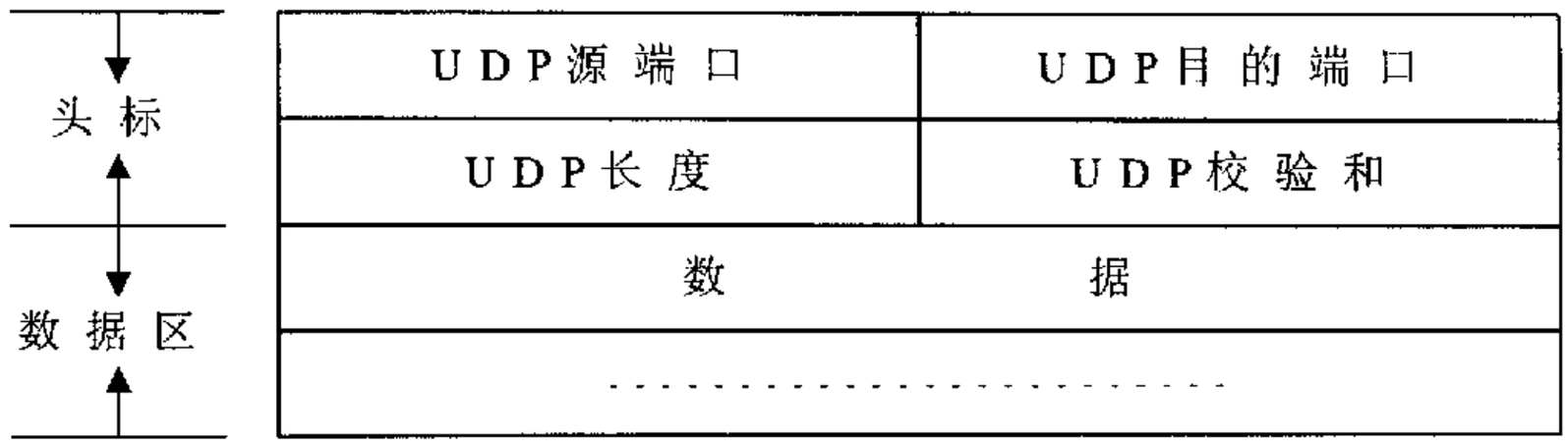

图 3-19 UDP 报文格式

UDP建立在IP之上，意味着UDP报文封装在IP数据报中传输。即发送端UDP程序模块将UDP报文交给IP程序模块后，IP程序模块在前面加一个头标，构成IP数据报：根据分层原则，接收端收到数据后，将对它进行一个与封装相反的过程，以保征对应层收到完全相同的数据。

2．UDP检验和与伪头标

UDP校验和是一个可选域，这是其效率的又一体现，因为计算校验和是很花费时间的，应用程序如果对效率的兴趣远远高于对可靠性的兴趣，就可以不选此域。但本论文的设计中要利用校验和来提高网络通信的可靠性，故选用校验和域。

UDP校验和还有一个与众不同的特点：除UDP数据报本身外，它还覆盖一个附加头标，该头标并非UDP数据报的有效部分，叫做伪头标（pseudo header）[2]。伪头标既不向下传送，也不向上递交，只在计算校验和时使用。伪头标格式如图3-20所示。

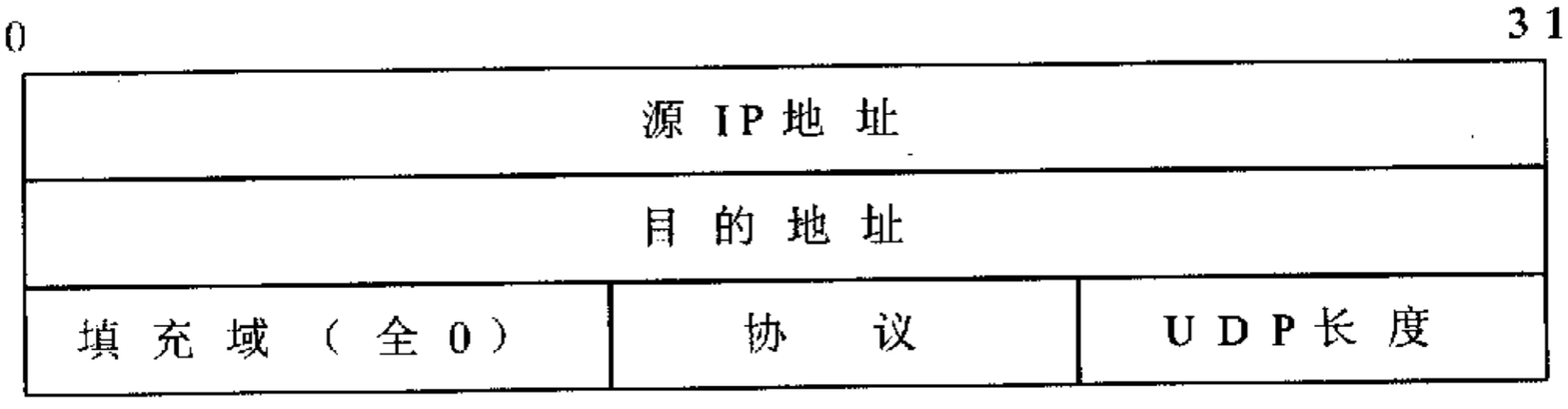

图 3-20 UDP 伪头标格式

UDP协议建立在IP协议之上，同IP协议一样提供无连接的数据报传输。相对IP协议而言，唯一增加的功能是提供协议端口以实现进程间的通信。用单片机实现UDP协议要做些简化，不考虑数据分片和优先权。

这一部分由三个函数组成：udp_rcve( )函数、udp_send( )函数、udp_echo_service( )函数。其中，udp_rcve( )函数负责解释、处理收到的UDP报文；udp_send( )

函数负责构成UDP报文并发送该报文；udp_echo_service( )函数负责把接收到的数据原样发送出去，主要是调试用，以后在应用中，数据处理函数将代替本函数。

发送数据时，上层的函数或本层的udp_echo_sevice( )函数调用本层的数据发送函数udp_send( )，在udp_send( )中再调用下层的ip_send( )函数实现发送。

当下层接收到数据时，调用本层的数据接收函数udp_rcve( )，在udp_rcve( )中通知用户接收任务有UDP数据报到达。

UDP协议的函数关系如图3-21所示。

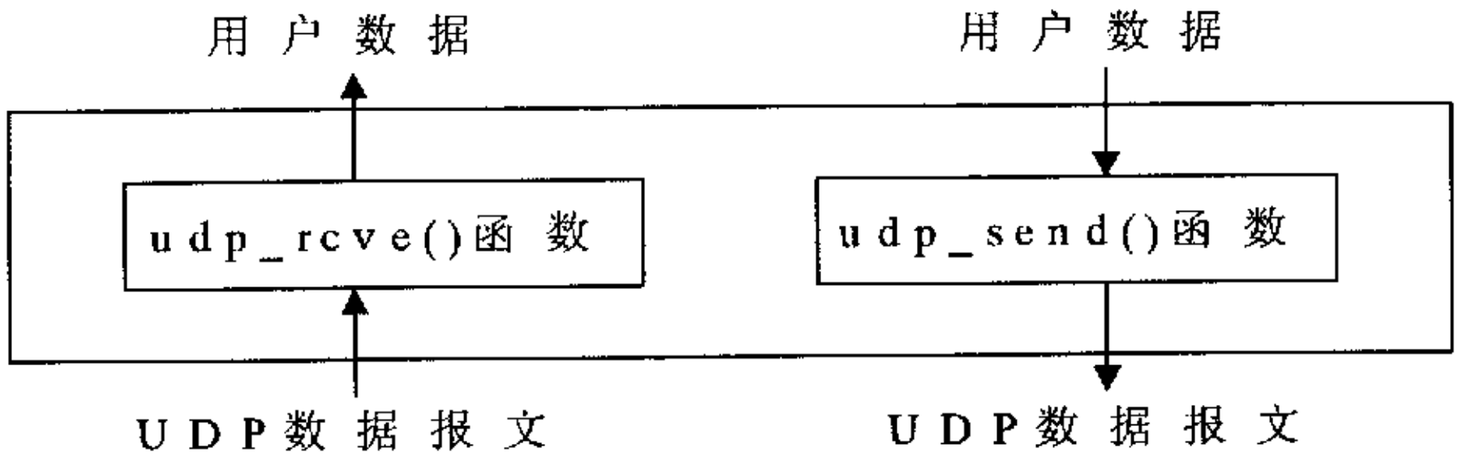

图 3-21 UDP 协议函数关系图

3.6.3 主要数据结构

UDP数据报首部结构：

```
typedef struct
{
    UINT   source_port;      /*源端口号*/
    UINT   dest_port;    /*目的端口号*/
    UINT   length;       /*数据报长度*/
    UINT   checksum;   /*UDP校验和*/
    UCHAR msg_data;  /*数据*/
} UDP_HEADER;
```

3.6.4 函数详细设计

1．udp_send( )：发送UDP数据报

封装UDP数据报，并调用ip_send( )函数将数据发送出去。udp_send( )主要完成UDP报头相关域的填写。为包括以太网首部和IP首部的完整的输出报文分配内

存。整个以太网的报文长度为：14字节以太网首部+20字节IP首部+8字节UDP首部+数据长度。流程图如3-22所示。

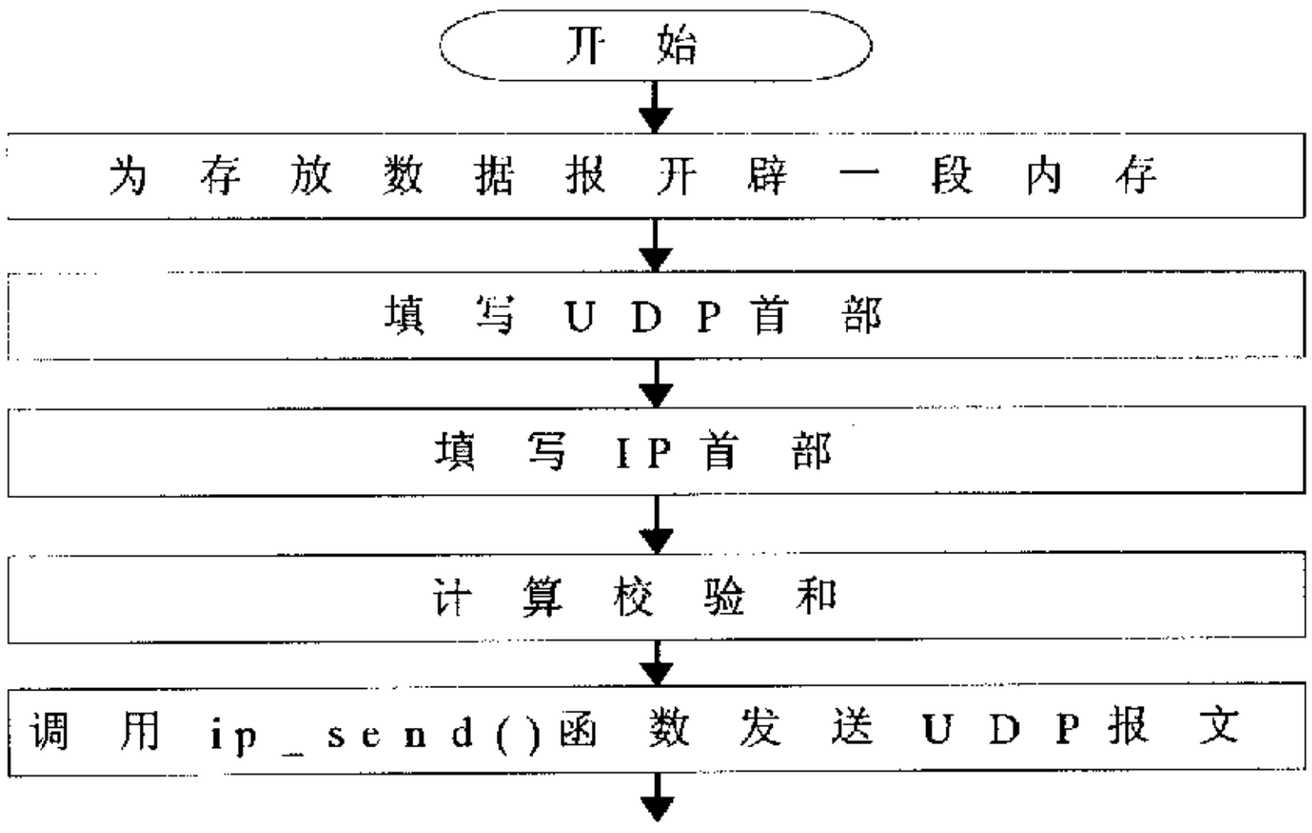

图 3-22 udp_send( )函数流程图

2. udp_rcve( ): 接收UDP数据报

重新计算校验和，判断是否接收。若校验和正确，将接收到的数据写入相应的端口；若校验和不正确，则抛弃该数据报。流程图如3-23所示。

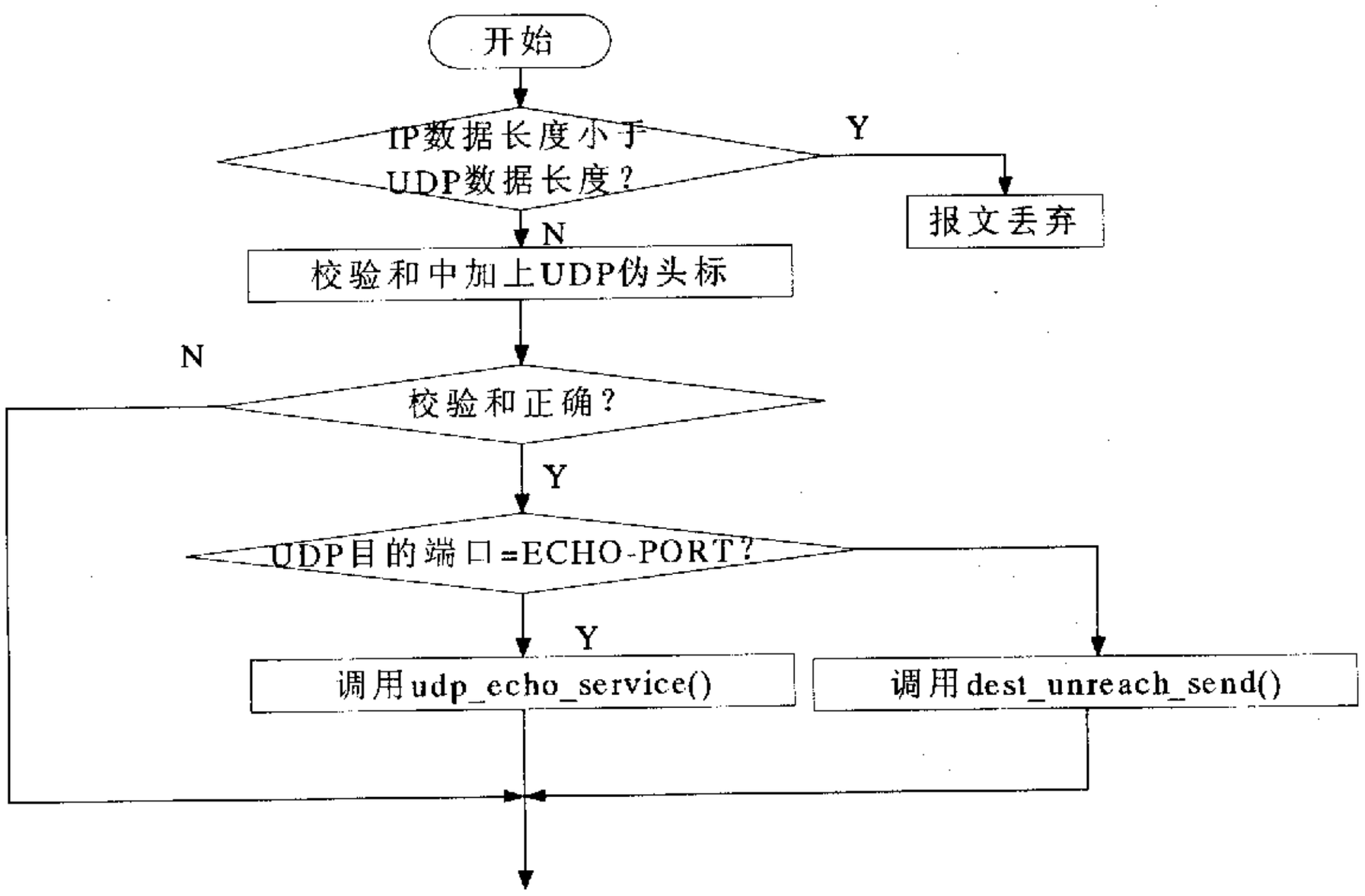

图 3-2 udp_rcve( )函数流程图

3．udp_echo_service( )：简单的处理UDP 协议的函数，将接收到的字符再发送回源IP地址

第 4 章 瘦服务器应用系统的实现与运行

4.1 瘦服务器功能实现与运行

用网线将一个瘦服务器接入局域网，打开局域网内的任意一台计算机，在计算机上做如下测试。

4.1.1 ICMP 协议程序的运行

假设瘦服务器的IP地址为192.168.0.10（IP地址定义在文件Main.c内），则在DOS提示符下输入PING 192.168.0.10 会观察到如图4-1结果。

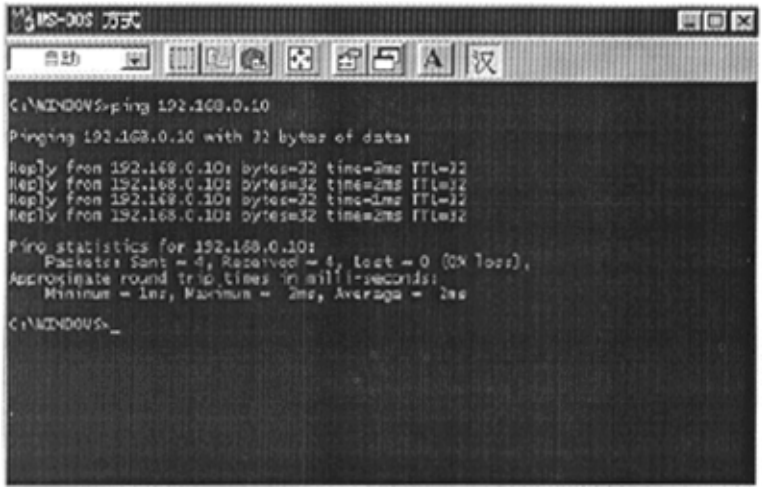

图 4-1 ICMP 协议测试窗口

此运行结果表明，本论文设计的瘦服务器实现了网络层及以下各层的协议和功能，局域网中各节点能在网络层上正常收发数据。

4.1.2 UDP 协议程序的运行

测试程序为udpdemo.exe，UDP的端口号为7（端口定义在文件Net.h内），执行udpdemo.exe程序窗口如图4-2。

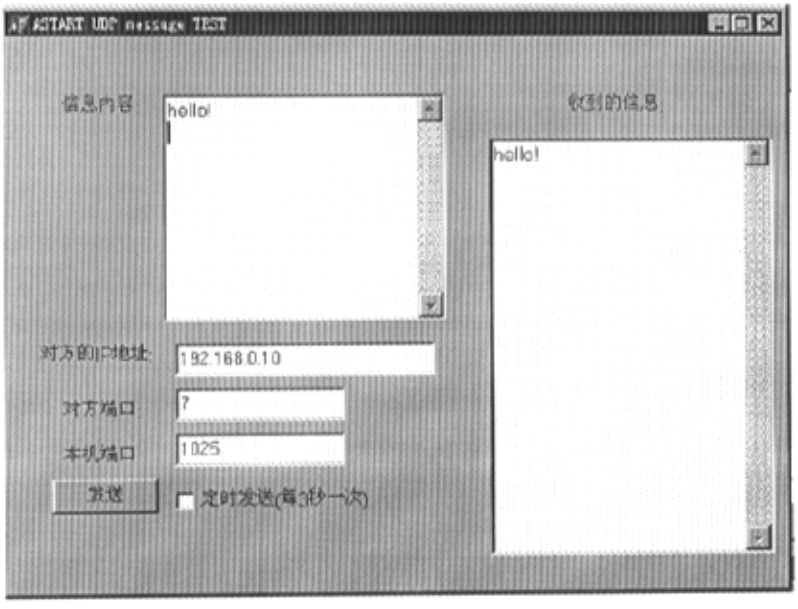

图 4-2 UDP 协议测试窗口

在信息内容编辑框内输入“hello!”，对方IP地址输入192.168.0.10，对方端口设为7，则按发送按钮，在收到的信息窗口会出现“hello!”。

此运行结果表明，本论文设计的瘦服务器实现了传输层及以下各层的协议和功能，局域网中各节点能在传输层上正常准确地收发数据。

4.2 瘦服务器应用系统的运行示例

由 LabVIEW 编写的瘦服务器应用系统中的客户端应用程序的前面板非常复杂，而且套色系统每种颜色的参数设定与误差显示都是类同的，所以下面我们以其中最具代表性的两个界面为例，详细说明系统的工作过程。

4.2.1 套色系统初期设定界面

本应用程序实现的是一个 9 色印刷控制系统，系统的初期设定界面如图 4-3 所示。

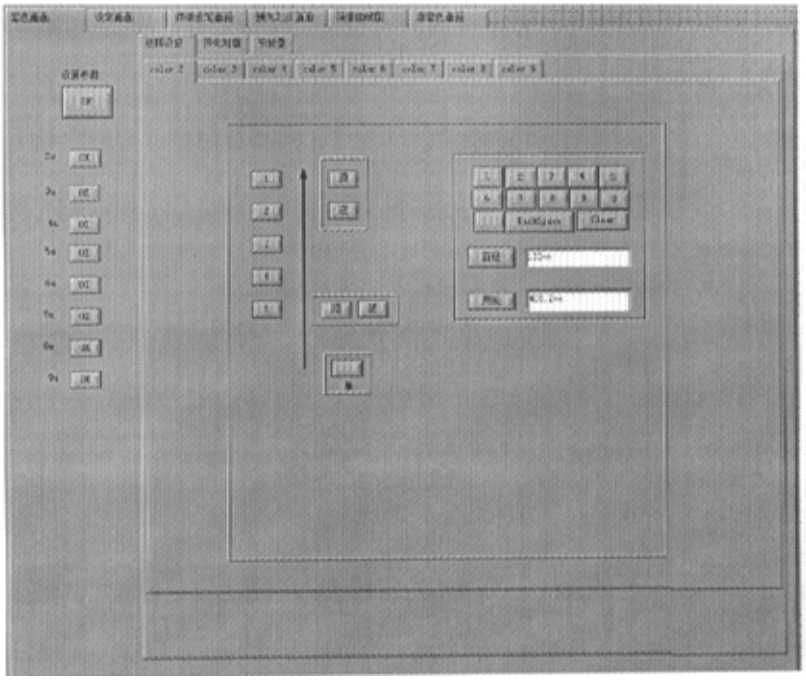

图 4-3 套色系统初期设定界面

1．界面介绍

这是“套色画面”菜单中第 2 色的初期设定界面。由于第 1 色印刷时无需对准，所以各项操作与控制都从第 2 色开始。可以看到，界面的最左边“设置参数”项下面共有 1 个大的、8 个小的“OK”按扭：小的“OK”按扭左边分别标上了“2μ”~“9μ”，它们表示印刷机的机组数；如果某个机组安装了版辊，将要进行印刷，就选择与其机组数对应的小的“OK”按扭；全部设置好了后，选择大的“OK”按扭确定以上的输入。界面中间的第 2 色初期设定界面中分为两个部分：色标选择和版辊周长设置。色标选择中间有一个向上的箭头，左边为色标形式选择，右边为色标排列顺序选择，都是用来设置第 2 色印刷时的一些参数的。版辊周长设置的上部分是一个输入键盘，可以通过触摸屏直接按压输入版辊的直径或是周长，只要输入其中任意一项，程序会自动计算出另一项。

2．数据流介绍

事实上，“设置参数”中每个小的“OK”按扭都记录着对应机组上安装的瘦

服务器的 IP 地址。本界面中，第 2 色的色标形式、色标排列顺序以及版辊周长一旦设置好后，就可以按照 TCP/IP 协议逐层封装这些参数，然后将封装好的数据包送入网络。网络中各个瘦服务器都一直处于等待状态，它们接收到该数据包后，会判断该数据包是否是发送给自己的。只有第 2 色上的瘦服务器会发现该数据包是给自己的，通过逐层解包数据而得到第 2 色的色标形式、色标排列顺序以及版辊周长，然后在用这些参数去控制印刷机。而其它的瘦服务器在处理数据包时会发现该数据包不是发给自己的，而将它丢弃。

4.2.2 套色系统波形显示界面

以上是客户端将设定好的命令或参数发送给系统中瘦服务器的过程。下面介绍瘦服务器将光电传感器采集到的数据传送给客户端，然后在客户端处理形成波形的过程。系统的波形显示界面如图 4-4 所示。

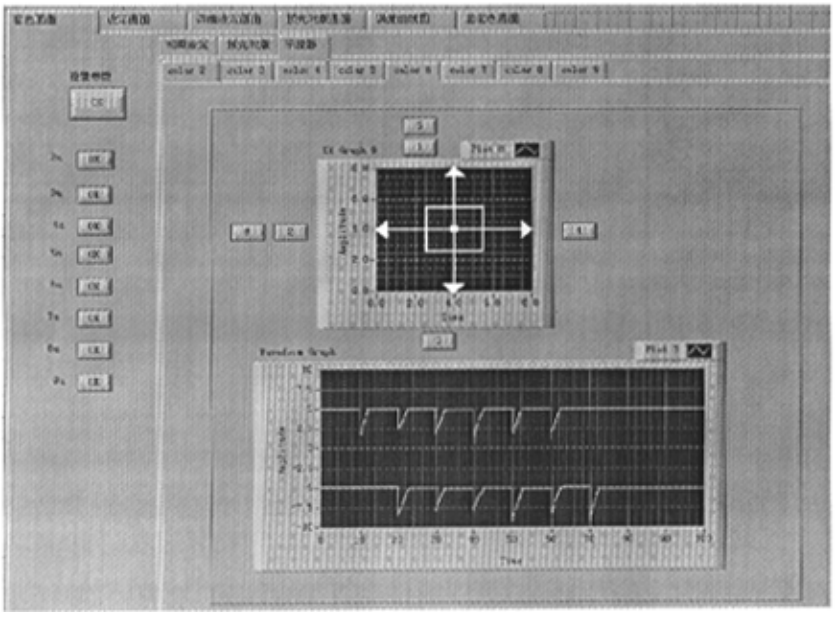

图 4-4 套色系统波形显示画面

1．界面介绍

这个界面是6色的波形显示画面，它主要由两个部分组成：上面正方形的是误差显示区，下面长方形的是波形显示区。

波形显示区显示实际印刷的色标信号。图4-4中下面的示波器是6色的波形显示画面，所以有6个色标信号，表现在图中就是两条有六个脉冲信号的波形。实际操作中，我们关心的是，上下两条波形的前后脉冲是否对准，如果如图中所示，上面一条波形的第n+1个脉冲与下面一条波形的第n个脉冲对准，则表示实际印刷中，印刷的各种颜色套准了。

误差显示区中我们可以看到有一个十字箭头，一个表示安全范围的方框和一个小方块。误差显示区中心的小方块偏离中心越远表示误差越大。如果小方块位于表示安全范围的方框外说明颜色没有套准，这时用户需要输入命令进行调整以免印刷次品；反之，如果小方块正好在十字箭头的交汇点上则说明颜色套准的误差为0，也就是颜色完全套准。

2. 数据流介绍

此界面中下面的波形显示区的数据来自于网络中的瘦服务器。图5-4中的波形一共有6个脉冲，说明这是与5个相应的瘦服务器通信的结果（印刷时，第一色无需对准，所以第一色上不安装瘦服务器）。在用户要求波形显示后，程序从图5-3的界面中获得有印刷动作的5个机组上安装的瘦服务器的IP地址，分别与这5个瘦服务器以TCP/IP协议通信，得到相应的色标信号。客户端程序得到这些信号后将数据放在一个数组中，然后在LabVIEW编写的示波器中完成波形的输出。另外，同样利用这个数组，程序还计算出第6色的纵、横向误差，形象地显示在误差显示区中。

本文所设计的瘦服务器已经通过网络测试和使用，运行良好，证明在实践中是可以使用的。该服务器硬件配置简单，软件界面友好，鼠标和触摸屏操作方便，可以根据需要完成不同的测试任务。另外，采用图形化编程语言LabVIEW开发的客户端应用程序程序快速、高效，界面与实际仪器面板非常形似，熟悉使用实际仪器的使用者可以通过触摸屏直接操作；波形的显示处理与网络功能的实现也比较简单。

以智能仪表和分散控制为特色的现场总线技术，把工业控制网络带入了一个新的时代。然而目前出现的多种现场总线没有一个统一的标准，各种现场总线产品之间很难实现互操作。本文设计的瘦服务器基于嵌入式Internet技术，其通信协议和通信介质都是通用的，而且利用智能仪器中的单片机，可以完成

整个工控网络和 Internet 的无缝连接。这样，不但使工控设备的远程控制和管理方式有了改变，而且大大降低了成本。

另外，通常情况下，网络中配置最好、内存最大、运算速度最快的单元会被作为服务器。但是，在本论文实现的印刷控制系统中，没有将工业 PC 机作为服务器，而是将印刷机控制单元中作为服务器。这是因为，一方面印刷机的机械速度远远低于单片机 CPU 的运算速度，而且需要通过网络传输的数据不多；另一方面，在印刷控制系统中，如果将工业 PC 机作为服务器，各个印刷控制单元作为客户机，则整个控制网络有一个服务器多个客户机，此时一旦服务器出现故障，很可能导致整个网络瘫痪；而如果将各个印刷控制单元作为服务器，工业 PC 机作为客户机，则整个控制网络有一个客户机多个服务器，此时即使客户机出现故障，服务器还能实现控制功能，不至于影响整个印刷机的工作。在印刷控制系统中，控制单元要负责印刷的实际操作，所以将它设计成服务器，可以防止因为工业 PC 机出现故障导致印刷机无法工作的情况。

实践证明，将本文设计的瘦服务器应用于印刷控制乃至其他工控系统中，能简化系统的结构，降低系统的成本，提高系统的兼容性和可靠性。

第5章 结论

5.1 主要研究工作总结

随着计算机技术、微电子技术和网络技术、软件技术的不断发展，世界主流计算机技术已进入后 PC 时代。后 PC 时代的来临加速了嵌入式系统向人们工作生活中的渗透，而包括工业控制现场应用的大量智能电子设备面临着愈加迫切的联网需求。

本文正是在这样的背景下开展了对嵌入式Internet技术的初步探讨。在对嵌入式Internet技术的概念、发展及其应用前景进行初步概括后，本文详细介绍了基于嵌入式Internet技术的瘦服务器的整体设计原理及硬件电路框图，接下来分析了在瘦服务器硬件平台基础上的嵌入式TCP/IP协议栈的实现和测试，另外，用LabVIEW实现的客户端应用程序，更代表了当前网络化虚拟仪器发展的新趋势。最后，通过一个实际的应用，设计了一个单客户端多服务器的控制网络，证明本文提出及实现的方案是可行的。这一实现方法的扩展和深化，对自动测控领域的测控网络化也有很好的借鉴价值。

将通用的网络通讯协议与嵌入式技术相结合，不仅需要对 TCP/IP 网络通讯协议有透彻的了解，而且需要对嵌入式领域的某些特点要求有非常准确的认识以及对嵌入式软件开发的过程和环境十分熟悉。只有对这三个方面的技术都有相当的掌握后，才能按照嵌入式应用需求，对通用的 TCP/IP 协议在不违背协议标准的前提下加以改进，从而满足嵌入式应用的要求。

为了项目课题开发的需要，我先后仔细研究了通用的TCP/IP协议标准和 4.4BSD TCP/IP内核实现的源代码，以及国外一些典型嵌入式网络软件产品。在参与本项目设计过程中，笔者主要作了如下工作：

（1）在仔细研究了 TCP/IP 协议标准和 4.4BSD TCP/IP 内核源代码的基础上，提出了一个基于嵌入式 Internet 的瘦服务器的实现模型，为今后的嵌入式 TCP/IP 研究打下了基础。同时，也为今后的改进工作建立了一个基本框架。

（2）在认真阅读、理解、消化各类芯片如 Cygnal（C8051F020）单片机、SPI 串行方式 8M 位的 Flash 存储器 AT45DB081、UT62L1024 SRAM、10M 以太网芯片 RTL8019AS 等相关资料基础上，完成相关硬件的调试工作。

（3）重点研究了瘦服务器中嵌入式 TCP/IP 协议栈的数据与实现，具体包括以太网协议、ARP 协议、ICMP 协议、IP 协议以及 UDP 协议。

（4）在嵌入式 TCP/IP 协议栈的基础上，实现了客户端应用程序。

（5）将研究成果成功的运用到了大型印刷机的套色控制系统中，并提出以印刷控制单元作为服务器，工业 PC 机作为客户机的单客户机多服务器的网络结构，实现了印刷机的远程控制和管理，网络配置简单，系统可靠性高。

5.2 有待解决的问题

但是，由于时间和能力有限，目前的工作还有一些不足之处，有待下一步工作的完善。总结一下，主要有如下几方面：

（1）在协议栈的实现中，只是初步实现了瘦服务器数据收发的基本功能，对其具体的性能如网络数据传输速度并没有进行定量的测试和分析，因此程序有进一步完善的必要；

（2）TCP和UDP都是建立在IP基础上的传输层协议。由于考虑到实际的应用，本文设计的瘦服务器只实现了UDP协议，而没有实现TCP协议。下一步可以实现TCP协议，进一步提高网络的可靠性和通用性。

（3）目前方案中单片机系统只提供了通过网卡芯片有线接入 Internet 的方式，下一步可以增加对 GPRS 无线模块的支持[49]。

（4）下一步将增加对 IPv6 协议的研究与实现。IPv6 将 IP 地址从 32 位扩大到 128 位，同时提供了诸如邻居发现等功能，这些功能对于嵌入式 Internet 而言更有意义[50]。

总之，通过课题设计，我在科研方面得到了极大的锻炼，培养了独立思考和实践的能力，我感觉到这段时光很充实，并且相信我所学和所研究的知识会让我受益终生。

限于作者水平有限，文中难免存在不完善的地方，恳请各位专家批评指正。

参考文献

[1] Douglas E. Comer. 用 TCP / IP 进行网际互连第 1 卷：原理、协议和体系结构（第 3 版）. 北京：电子工业出版社，1998.4

[2] W. Richard Stevens. TCP/IP详解 卷1：协议. 北京：机械工业出版社，2000.4

[3] Jacek W. Szymanski. Embedded Internet Technology In Processing Control Device. 2000 IEEE：301～308

[4] Burton H. Lee. Embedded Internet System: Poised For Takeoff IEEE Internet Computing 1998：24～29

[5] Robert E. Filman. Embedded Internet System Come Home IEEE Internet Computing. 2001：52～53

[6] Janne，Jussi Roivainen. Providing Network Connectivity For Small Applications：A Functionally Minimized Embedded Web Server IEEE Communications Magzine 2001 October：74～79

[7] NS Manju Nath. LOW-COST Techiques Bring Internet Connectivity to Embedded Device. EDN. 1999：159～166

[8] 马义德，刘映杰，张新国. 嵌入式系统的现状及发展前景. 信息技术，2001（12）：57～59

[9] 赵葵银，唐勇奇. MCU 应用系统与 Internet 连接的一种新技术. 单片机与嵌入式系统应用，2001（2）：20～23

[10] 吕京建. 嵌入式因特网技术的兴起与前景. 今日电子，2000 增刊：16～18

[11] 吴晓蓉等. 互连网技术在嵌入式系统中的实现. 计算机工程，2001.4

[12] 沈绪榜. 嵌入式系统与芯片技术. 嵌入式系统论文集，1～3

[13] 绍兴军，吕京建. 嵌入式微型互联网及其应用研究. 嵌入式系统论文集，31～35

[14] 窦振中. 嵌入式系统设计方法的演化. 单片机与嵌入式系统应用，2001（1-6）合订本，3～6

[15] Shear，David. Putting an Embedded System on the InternetEDN. 1997，9：37～46

[16] Thomas F. An Introduction to TCP/IP for Embedded Engineers Embedded System Coference，SanFrancisco 2002：350～370

[17] J. Benthan. TCP / IP Lean：Web servers for embedded system CMP Books，2000：15～20

[18] 杨志红等. 基于嵌入式网络技术构建远程监测系统. 测控技术，2002. 8

[19] RTL8019AS Realtek Full-Duplex Ethernet Controller with Plug and Play Function REALTEK SEMI-CONDUCTOR CO., LTD. 2000：3～25

[20] Philip J. Koopman. Embedded System Design Issues. Proceedings of the International Conference on Computer Design（ICCD 96）

[21] J. Postel. Internet protocol RFC791. Internet Engineering Task Force，September 1981：9～33

[22] R. Braden. Requirements for Internet Hosts－－Communication Layers. RFC1122. Internet Engineering Task Force，October 1981：29～38

[23] J. Postel. Transmission control protocol RFC793. Internet Engineering Task Force，September 1981：14～84

[24] Charles Horning. A Standard for the Transmission of IP Datagrans over Ethernet Networks RFC894. Internet Engineering Task Force，April 1984：1～2

[25] J.Postel，J. Reynolds. A Standard for the Transmission of IP Datagrans over IEEE 802 Networks RFC1042. Internet Engineering Task Force，February 1988：1～14

[26] Anthony Jones. Network Programming for Microsoft. 北京：机械工业出版社，2000：132～340

[27] J. Postel. User datagram protocol RFC768. Internet Engineering Task force，Augest 1980：1～3

[28] Douglas E.Comer，David L.Stevens. 用 TCP/IP 进行网际互联 第二卷：设计、实现与内核（第三版）.北京：电于工业出版社，2001.4

[29] Gary R.Wright，W.Richard Stevens. TCP/IP 详解 卷 2：实现. 北京：机械工业出版社，2000.7

[30] 程军. 单片机远程控制方案及基于互联网的实现方法. 微计算机信息. 2001. 17（1）：47～50

[31] RTL8019AS Specification. Realtek Semi-conductor CO., LTD.

[32] 徐爱钧，彭秀华. 单片机高级语言 C51 Windows 环境编程与应用. 北京：电子工业出版社，2001. 122～327

[33] 刘彦明，李鹏. 实用网络编程技术.西安：西安电子科技大学出版社，1998. 12～54

[34] 葛永明，林继宝. 嵌入式系统以太网接口的设计. 电子技术应用，2002（3）：25～27

[35] 汪蒲阳. 因特网应用编程. 北京：清华大学出版社，2000. 271～300

[36] 陈坚，孙志月. Modem 通信编程技术. 西安：西安电子科技大学出版社，1999. 41～91

[37] 梁合庆. 关于 Internet 网和 TCP/IP 协议的实用技术. 电子产品世界，2000（7）：57～58

[38] 许华杰，明健. 基于 SX 单片机实现 Webserver 和网络协议栈. 单片机与嵌入式系统应用，2001（11）：28～30

[39] Quninell，Richard A. Web Servers in Embedded System Enhance User Interaction. EDN，1997. 61～63

[40] 马忠梅. 嵌入式应用设计模式. 单片机与嵌入式系统应用，2001（1-6）合订本，30～32

[41] 王世昌. 计算机系统与网络技术开发及应用实例（二）. 北京：机械工业山版社，1998.4

[42] LabVIEW User Manual. National Instruments Corp.，1999

[43] 杨乐平等. LabVIEW 程序设计与应用. 北京：电子工业出版社，2001

[44] 裘伟廷， 基于 LabVIEW 的虚拟仪器和虚拟实验。 现代科学仪器，2002（3）.3

[45] G Programming Reference Manual. National Instruments Corp.，1999

[46] 约翰逊. G. W. LabVIEW 图形编程. 北京：北京大学出版社，2002.

[47] 李铁等，基于 LabVIEW 的虚拟仪器技术在无损检测中的应用，无损检测，2001（6）

[48] 赵伟，张小牛，孟浩文. 网络化——测量技术与仪器发展的新趋势. 电测与仪表，2000.37（7）：5～9

[49] 何小庆. 嵌入式实时操作系统的现状和未来. 单片机与嵌入式系统应用，2001（3）：8～10

[50] Morgan Kaufmann. IPv6 Clearly Exnlained. 北京：机械工业出版社，2000. 33～40

作者在攻读硕士学位期间发表的论文

[1] 周申培，陈明. MC68HC908GP32IDK 在键盘操作装置中的应用. 三峡大学学报(自然科学版)，2003. 4：331-333

[2] 周申培，周伟，李信. 利用 Win32API 开发串行通信程序. 计算机与信息技术，2003. 10:38-40

致　谢

我首先要感谢导师周伟老师，在近三年求学期间，无论在生活还是学习上，周老师都给予了我足够的关心和支持，他的支持和鼓励使我能够顺利完成本论文的设计。周老师渊博的学识、严谨的治学作风、敏锐超前的学术意识、对科学研究全身心的投入、认真负责的工作态度，使我感受很深、受益匪浅。在整个项目设计和论文编撰过程中，周老师帮助我拓宽了思路，提高了能力。在此，我对周老师表示诚挚的谢意！

另外，还要感谢武汉楚鹰科技发展有限公司各位老师、同事给予我的无私的帮助与支持。同时，我还要感谢我的老师、同学、朋友，他们在我攻读硕士期间给予了我很多关心与照顾；我要特别感谢我的家人，正因为他们的支持我才能够顺利地完成学业。最后，我要对所有关心过我、帮助过我的人表示最衷心的谢意！