中文摘要

数字电视机顶盒是当今发展最快、研究最多的热门技术，基于Internet的业务是数字电视机顶盒非常重要的应用。

目前，数字电视机顶盒的核心控制处理器主要有单片机和嵌入式系统两种解决方案。针对核心控制器为单片机的情况，本论文实现了TCP/IP协议栈在核心控制器上的移植。论文采用单片机开发系统和PC机构成了实验环境，在单片机端的调试软件采用Keil开发软件；PC机上面的软件采用的是Visual Basic 6.0。具体而言，即首先PC机通过串口发送测试数据给单片机；单片机则通过以太网将数据回传送给PC机；PC机负责检验并显示接收到的数据并给出检验报告。实验证明，本设计可实现了以太网包的收发，ping等命令。

对于采用嵌入式系统的机顶盒，本论文选择意法半导体公司(ST)推出的数字电视机顶盒解决方案STx5105平台，提出了基于网卡芯片DM9000A移植LWIP的网络解决方案。LWIP是TCP/IP协议栈的一种实现，即Light-Weight TCP/IP 协议栈，其主要目的是减少存储器利用量和代码尺寸，使LWIP适合应用于小的、资源有限的处理器如嵌入式系统。LWIP有几个模块组成，除了实现TCP/IP协议的各个模块(IP、ICMP、UDP和TCP)之外，还同时设计了许多支持模块。这些支持模块组成了操作系统模拟层、缓冲和存储管理子系统、网络接口函数等。LWIP采取的是把所有协议封装到一个单一的过程中，实现与操作系统内核分开，从而便于在不同的操作系统上移植。本文详细介绍了OS20操作系统，并详细阐述了LWIP如何与OS20操作系统、应用层以及驱动层的封装问题。

关键词： 数字电视机顶盒　TCP/IP　LWIP 移植　OS20 实时操作系统　STx5105

ABSTRACT

Digital TV set-top box is the fastest growing and most popular technology research recently. Internet-based digital TV set-top box business is one of the most important applications.

At present, the main processors of digital TV set-top boxes have two solutions, the MCU and the embedded systems. For the situation of SCM as the main processor, this paper realized a TCPIP stack on it. This paper uses the environment of the SCM system and the PC system. The SCM client software is Keil. The computer software is Visual Basic 6.0. Particularly, PC sends data through the serial port to SCM, MCU transmitted the data to the computer via ethernet, PC is responsible for testing and displaying the data received and gives the inspection reports. Experiments show that Ethernet transceiver package and ping command can be successfully performed.

For the set-top boxes of embedded operating systems, this paper chooses STx5105 as the platform which is the solution of the company of ST, this paper realized a LWIP transplant network solution based on the network chip card DM9000A. LWIP is one kind of TCP / IP protocol which means Light-Weight TCP/IP stack. The main purpose of LWIP is to reduce the use of memory, so LWIP suites for small, limited resources such as embedded system processors. There are several modules in LWIP, in addition to TCP/IP basic modules (IP, ICMP, UDP, and TCP), it also contains supporting modules. These modules contain operating system simulation layer, buffer and storage subsystem, several network interface functions and so on. In order to facilitate the transplant to different operating system, LWIP packages all protocols into a single process, with the core operating system separately. The paper described the OS20 operating system, and elaborated on the packaging problem between LWIP and the OS20 operating system, the application layer and the driver layer.

KEY WORDS: Digital TV set-top boxes , TCP/IP , LWIP Transplant , OS20 Real-time operating system , STx5105

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：高新立　　签字日期：2008 年 6 月 5 日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：高新立　　导师签名：刘开华

签字日期：2008 年 6 月 5 日　　签字日期：2008 年 6 月 5 日

第一章 绪论

1.1 引言

数字电视是指从演播室到发射、传输、接收的所有环节都是使用数字电视信号或对该系统所有的信号传播都是通过由0、1数字串所构成的数字流来传播的，数字信号的传播速率是每秒19.39兆字节，如此大的数据流的传递保证了数字电视的高清晰度，克服了模拟电视的先天不足，而且相对于模拟电视，数字电视在各方面都具有压倒性的优势，无论是国内还是国外，电视数字化已经成为不可逆转的趋势。

与传统的模拟电视相比，数字电视的优点主要体现在：

第一，提高了频率资源的利用率。利用数字压缩技术可以在一个标准有线电视模拟频道中传输4—10套电视节目。

第二，提高电视信号的传输和接收质量，可以保证用户接收到和前端播出效果基本相同的电视信号。

第三，可以提供数据广播。

第四，逐步改变观众传统的收视习惯，由被动收看到准视频点播(NVOD)收看，以至下一步收看真正的视频点播(VOD)[1]。

节省频率资源，有利于节目数量的增加和频道的专业化，可满足不同观众群体的需要；在目前模拟电视资源渐渐饱和的情况下，电视的数字化对于电子厂商、广播电视业者来说更是一个难得的机遇。我国将在2008年全面推进数字高清晰度电视，2010年基本实现数字化，2015年停止模拟信号的播出。

1.2 数字电视机顶盒的分类

目前数字电视主要有两种标准。一是欧洲 ETSI 的 DVB(日本 DiBEG 的 ISDB-T 源于 DVB，不另作分类)；二是美国先进电视委员会 ATSC 的 DTV。

DVB 家族分为三个部分：用于卫星数字电视广播的 DVB-S；用于有线(同轴电缆)数字电视广播的 DVB-C；以及用于地面数字电视广播的 DVB-T。其中 DVB-S 标准已为全球所认同；DVB-C 为欧洲、澳大利亚、北美、南美等一些国

家接受；而数字电视地面广播 DVB-T 已在欧洲、澳大利亚、新加坡进行了广泛的测试试验得到认可。

DVB-S 用于卫星信道。卫星信道的特点是：可用频带宽、功率受限、干扰大、信噪比低。所以要求采用可靠性高的信号调制方式、强的信号纠错能力，对带宽要求不是特别高。因此 DVB-S 采用前向纠错(FEC)(包括 Viterbi 编码、交织、RS 编码及加扰等电路)，正交移相键控(QPSK)调制的信道处理,然后馈给卫星链路。接收时进行相反的处理。

DVB-C 用于有线信道。有线信道的特点是：信噪比高、频带资源窄、存在回波和非线性失真。这些特点要求 DVB-C 采用带宽窄、频带利用率高、抗干扰能力较强的调制方式。同时，由于信道信噪比高、误码率较低，纠错能力要求不很高。因此，DVB-C 的信道部分采用 RS 码和卷积码交织技术，正交幅度调制(QAM)。

DVB-T 用于地面广播信道。地面广播的特点是：地形复杂、存在时变衰落和存在多径干扰、信噪比较低。因此 DVB-T 采用前向纠错(FEC)(包括内码交织、内码 Viterbi 编码、外码交织、外码 RS 编码)和能有效消除多径干扰的正交频分复用技术(COFDM)和格雷码映射 4/16/64QAM 调制等进行信道处理。然后在原来用于模拟的 6MHZ、7MHZ 和 8MHZ 的频带内发送数字电视节目。DVB-T 发送的比特率是可变的。例如：在 6MHZ 频带可在 3.7～23.8Mbit/s 比特率之间进行选择；在 8MHZ 频带可在 4.9～31.7Mbit/s 比特率之间进行选择。以适应不同的接收环境、如移动接收应适当降低发送的码率[2]。

ATSC 的 DTV 是一种地面数字电视广播标准，与 DVB-T 形成竞争，已在澳大利亚、新加坡等国家与 DVB-T 进行对比试验。目前接受该标准的国家和地区有美国、加拿大、墨西哥、阿根廷、韩国、台湾等。另外，北美地区在卫星数字电视广播方面接受 DVB-S、DSS(休斯数字卫星系统)；在有线数字电视广播方面接受 OpenCable(美国 CableLabs 制定的数字有线标准，该标准接受 ATSC 制式以及国际电讯联盟（ITU）的 ITU-TJ.83 的用于电视、声音和数据服务的有线数字多节目制式)。

1.3 数字电视接入 Internet

近几年数字电视的迅速发展,如何在数字电视机顶盒的嵌入式系统中实现和Internet的连接,实现有线电视网络和Internet的初步融合以及数据之间交流,成了近年来的热点。随着互联网技术的发展和普及,三网(电信网、计算机网和有线电

视网)融合的新时代已经来临。上世纪末还只是单一的计算机网络大发展大繁荣时期,而本世纪将是综合了多种业务的交互式宽带网络舞台,而信息家电将是宽带交互式新型多媒体家电终端的总称。它一方面继承了家电的全部功能特性,同时又与Internet相连，提供信息交流和处理等多种交互式业务。信息家电是数字音/视频信息处理技术、电信技术、计算机技术和有线电视技术相融合的产物，它以多种形式出现，在我国当前应用最普遍，分布最广泛的就是以电视接收机为终端的机顶盒技术。随着有线广播电视网向数字化、网络化、产业化方向发展，最终将建成宽带综合信息网，有线数字电视机顶盒成为依托有线广播电视网提供综合信息业务的关键设备之一[3]。数字电视机顶盒基于Internet的应用已经成为发展最快,研究最多的热门技术。

网络数字电视由于突破传统电视模式，克服现有电视频道受地区及气候等多种因素约束的弊病，跨越时间和空间约束，在网上实现无限频道的电视收视，成为目前信息技术和媒体产业的一个热点。以IP为基础的宽带网络发展迅速，网络承载能力不断增强，这必然会对业务的数量和质量提出更高的要求。因此，若能将数字电视网络上面的巨大媒体资源引入到IP网上，一方面可以丰富IP宽带网上的业务量，促进IP宽带网络的普及和发展，另一方面，网络平台的扩充，将给节目运营商带来更多的收视用户，从而可以降低成本，增加收入。

IPTV是传统电视与宽带网络的结合体,它通过引入IP机顶盒并把数字电视信号从IP网络传送到用户终端，将电视和Internet完美地结合起来，彻底改变了传统电视节目的单向传输方式，使观众和电视之间形成了一种互动的和谐关系，改变了过去人们被动地接受信息的状况，使人们可以主动的选取信息。近年来，IPTV在世界各国得到了迅速的发展，各大有线电视运营商及电信运营商都投入巨资进行这方面的研究及实验。

双向机顶盒的功能主要是既可以接收卫星、有线电视高清节目，同时具备DVB IP数据包的解包，流媒体的录制、回放和管理，DVD光盘播放，网页浏览，以及上行internet网络交互等功能。

基于Internet的业务可能是有线数字电视机顶盒最重要的应用。目前市场上的机顶盒能实现上网的有网络电视机顶盒和多媒体机顶盒。网络机顶盒的主要功能是使我国现有的数亿台模拟电视通过PSTN或双向CATV实现Internet接入。多媒体机顶盒通过双向CATV网可以支持所有的广播和交互式多媒体应用。

Internet业务是数字STB未来最重要的业务类型，STB的Internet接入是通过连接前端的文件服务器和有线电视网络实现的，文件服务器采用HTTP协议侦听和响应浏览器的请求。为了有助于浏览网络，STB通常要和代理服务器配合使用，代理服务器相当于网关，它在电视网和Internet之间设立防火墙类型的安全机制，

同时提供易于使用的管理特性。代理服务器还提供分布式超高速缓存，可以使网络效率提升一倍。

1.4 本论文的工作内容

有线数字电视机顶盒（STB）是信息家电中发展最快、研究最多的热门技术。基于 Internet 的业务是 STB 最重要的应用类型。本论文在详细论述了 TCPIP 参考模型和 OSI 参考模型并且分别介绍了各层的原理及实现之后，重点阐述了 STB 基于 Internet 的应用和实现。

对于以单片机为核心控制芯片的机顶盒产品，本论文完成了以下工作：

- 为单片机外接 RTL8019 网卡芯片，完成网卡芯片的硬件电路设计；
- 给单片机移植 TCP/IP 协议栈，实现了 ping 命令，ARP 中实现了缓存（学习、更新、老化、轮转替换）；
- 实现了 TCP、UDP 协议，实现了 PC 端与单片机通信的功能，及串口转以太网的功能。

对于 DVB-C 有线机顶盒基于 Internet 的应用，本论文针对 CPU 为 STx5105 操作系统为 OS20 的 ST 公司的解决方案，作了如下的研究和探索：

- 为 STx5105 外接 DM9000A 网卡芯片，参与完成网卡芯片的选型与硬件电路的设计与连接；
- 研究 STx5105 软件平台的整体架构，掌握其编译流程和运行方法；学习 OS20 操作系统以及 makefile 文件的编写；
- 学习 LWIP 协议栈的设计与移植，设计完成与操作系统，应用层和驱动层的封装问题。

第二章 TCP/IP 体系结构

2.1 OSI 网络参考模型

为了使局域网内部与局域网之间的通信能有一个统一的标准，国际标准化组织（International Standard Organization，ISO）在 1978 年提出了一个共同的网络通信参考模型，称为开放系统互连模型（Open System Interconnect，OSI），共包含七层,分别是:物理层，数据链路层，网络层，传输层，会话层，表示层和应用层,如图 2-1 所示。七层模型主要是提供分层管理的公用沟通模型,使两系统之间数据的通信(传送，接收，中断等)能更加容易管理，OSI 的各层都有其主要的功能,需要更新功能时，可以个别的修改某些层即可，不必修改整个构架。

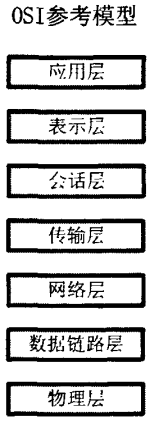

图 2-1　　OSI 七层模型

OSI 参考模型的最低三层（物理层、数据链路层、网络层），主要用来管理物理网络，偏向硬件技术，包含了各种规范如不同的通信介质：双绞线、同轴电缆、光纤等，不同的网络接入方式：载波侦听多路访问/冲突检测、令牌总线、令牌环等，以及不同的网络拓扑方式：星型、环型、总线等。

OSI 参考模型的高三层（会话层、表示层、应用层），是偏向软件技术，包含了字符的转换、数据的压缩及解压缩、用户应用程序等。OSI 参考模型的第四层传输层，主要是作为最高三层与低三层之间的沟通接口，确保传输的质量正常。

2.1.1 OSI 参考模型七个层次的介绍

物理层是 OSI 的第一层，它处于最底层，却是整个开放系统的基础。物理层负责数据位在物理传输媒体上的传输，使电气信号可以在两个设备间交换，主要包含网络的电气规范，如电压、电流的范围、连接器种类、连接器引脚定义、交换控制电路、传输速度、传输距离等。物理层的主要功能是为数据端设备提供传送数据的通路，数据通路可以是一个物理媒体，也可以是多个物理媒体连接而成。

数据链路层负责确保物理层连接的数据其正确性，包含数据传输的错误检验及错误更正功能。由数据链路层建立一个可靠的通信协议接口，使第三层网络层能正确地访问物理层的数据。

在常见的 IEEE 802 系列标准中，将数据链路层分为两部分：（1）逻辑链接控制（Logical Link Control，LLC）子层；（2）媒体访问控制（Medium Access Control，MAC）子层。

其中 MAC 子层是制定如何使用传输媒体的通信协议，如 IEEE 802.3 以太网标准的 CSMA/CD 协议中，MAC 子层规定如何在总线型网络结构下使用传输媒体；IEEE 802.4 令牌总线（Token-Bus）标准中，MAC 子层规定了如何在总线的网络结构下利用讯标（Token）控制传输媒体的使用；IEEE 802.5 令牌环（Token-Ring）标准中，MAC 子层规定了如何在环状网络结构下利用讯标来控制传输媒体的使用；IEEE 802.11 无线局域网标准中，MAC 子层规定如何在无线局域网络的结构下控制传输媒体的使用。

LLC 子层的主要工作是控制信号交换、数据流量控制（Data Flow Control），解释上层通信协议传来的命令并且产生响应，以及克服数据在传送的过程中所可能发生的种种问题（如数据发生错误，重复收到相同的数据，接收数据的顺序与传送的顺序不符等）。在 LLC 子层方面，IEEE 802 系列标准中只制定了一种标准，各种不同的 MAC 都使用相同的 LLC 子层通信标准，使更高层的通信协议可不依赖局域网络的实际架构。

网络层可以管理节点到另一个节点的路由，负责建立、维护及终止两个用户之间的连接，使数据根据其路由传输，因此该层必须提供寻址的能力。在网络层中数据的传输是以分组的形式来运作，所谓分组是指一组位数据，其包含了传送端、接收端节点地址位以及数据位。

传输层主要是确保数据在网络层与会话层之间的传输质量，即传送的数据正确，没有遗失，没有重复。传输层协议为不同主机上的应用程序进程提供逻辑通信。逻辑通信的意思就是尽管通信的应用进程之间不是物理连接的，而从应用

程序的角度来看，它们就像是物理连接的一样。应用程序通过使用传输层提供的逻辑通信互相传输信息，而不用考虑用来传送这些信息的物理基础设施。

计算机网络可以为网络应用程序制定多个传输层协议。例如，因特网(Internet)有两个协议——TCP 和 UDP。除了多路复用移路分解服务之外，传输层协议还可以给应用进程提供其他服务，包括可靠数据传输、带宽保证和传输延迟保证。

会话层主要在管理各个用户之间数据的交换形式。交换形式有单工、半双工及全双工。所谓单工，数据之传送与接收，只能单向操作。半双工，传送和接收数据可双向操作，但必须分开进行。而全双工，则可以双向同时传送和接收数据。

表示层负责将传输的信息以有意义的形式表达给网络的用户，其中包含了字符的转换、字符的编码与解码、数据格式的变换、数据的压缩与解压缩。

应用层是 OSI 模型的最高层，并提供了用户网络上的服务，如分布式数据库的访问、电子邮件、模拟终端机等。应用层为操作系统或网络应用程序提供访问网络服务的接口。应用层协议的代表包括：Telnet、FTP、HTTP、SNMP 等。

2.1.2 OSI 参考模型数据的封装过程

下面介绍一下 OSI 参考模型中的数据封装过程，如图 2-2 所示：

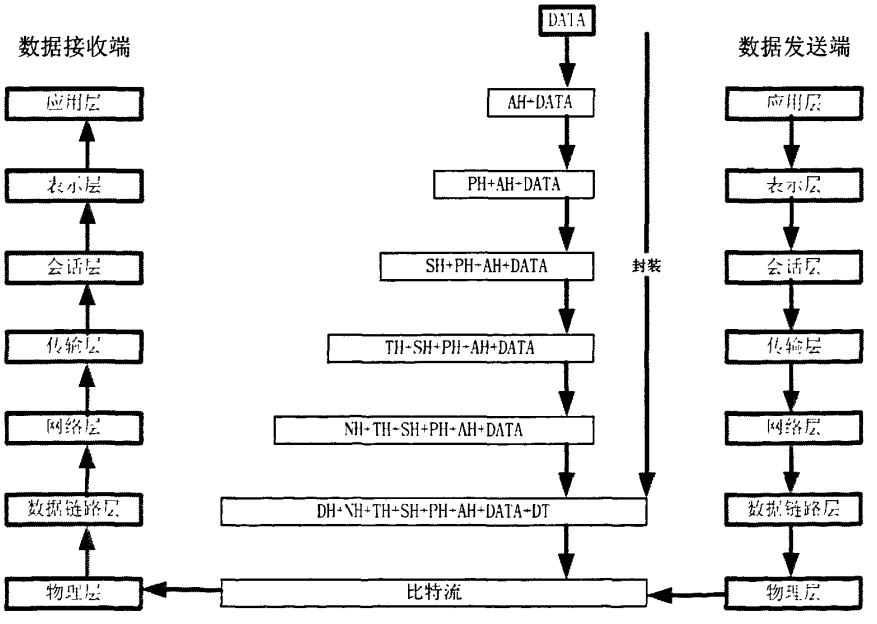

图 2-2　OSI 参考模型中的数据封装过程

如图 2-2 所示，在 OSI 参考模型中，当一台主机需要传送用户的数据（DATA）时，数据首先通过应用层的接口进入应用层。在应用层，用户的数据被加上应用层的报头（Application Header，AH），形成应用层协议数据单元（Protocol Data Unit，PDU），然后被递交到下一层-表示层。

表示层并不“关心”上层-应用层的数据格式而是把整个应用层递交的数据包看成是一个整体进行封装，即加上表示层的报头（Presentation Header，PH）。然后，递交到下层-会话层。

同样，会话层、传输层、网络层、数据链路层也都要分别给上层递交下来的数据加上自己的报头。它们是：会话层报头（Session Header，SH）、传输层报头（Transport Header，TH）、网络层报头（Network Header，NH）和数据链路层报头（Data link Header，DH）。其中，数据链路层还要给网络层递交的数据加上数据链路层报尾（Data link Termination，DT）形成最终的一帧数据。

当一帧数据通过物理层传送到目标主机的物理层时，该主机的物理层把它递交到上层。数据链路层负责去掉数据帧的帧头部 DH 和尾部 DT。如果数据没有出错，则递交到上层-网络层。同样，网络层、传输层、会话层、表示层、应用层也要做类似的工作。最终，原始数据被递交到目标主机的具体应用程序中。

在 OSI 模型中，每一层都各司其职，各个层次都需要了解彼此对应层次的规范及法则，这就叫做协议，并且在这七层的模块中，下层必须提供服务给上层，两层之间的沟通方式，我们称之为接口，通过不同的接口数据可以下传至传输介质，而传输介质也通过相反顺序的接口，上送到另一个工作站，这就是 OSI 模型的主要的功能，即数据的下传与上送。

OSI 模型的制定比 TCP/IP 晚了近 10 年，而两者之间相同的部分是两者都采用分层管理的方式。

2.2 TCP/IP 网络模型

2.2.1 TCP/IP 与 OSI 参考模型的对比

OSI 参考模型过于庞大招致许多批评，由技术人员开发的 TCP/IP 协议栈获得了广泛的应用。在 TCP/IP 参考模型中，TCP/IP 参考模型的应用层与 OSI 参考模型的应用层相对应，TCP/IP 参考模型的传输层与 OSI 参考模型的传输层相对应，TCP/IP 参考模型的互联网络层与 OSI 参考模型的网络层相对应，TCP/IP 参考模

型的主机-网络层与 OSI 参考模型的数据链路层和物理层相对应。在 TCP/IP 参考模型中，对 OSI 参考模型的表示层、会话层没有对应的协议，如图 2-3 所示：

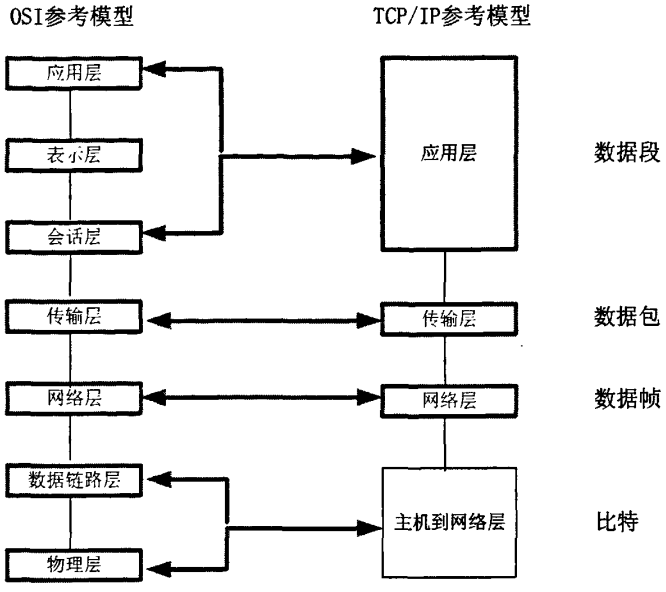

图 2-3 TCP/IP 参考模型

TCP/IP 协议栈是美国国防部高级研究计划局计算机网（Advanced Research Projects Agency Network，ARPANET）和其后继因特网使用的参考模型[4]。

2.2.2 TCP/IP 参考模型四个层次介绍

TCP/IP 模型分为如下四个层次，如图 2-4 所示：

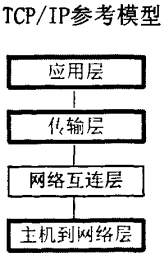

图 2-4 TCP/IP 参考模型的层次结构

下面分别介绍各层的功能:

(1)主机-网络层

在 TCP/IP 参考模型中，主机-网络层是参考模型的最低层，实际上 TCP/IP 参考模型没有真正描述这一层的实现，只是要求能够提供给其上层-网络互连层一个访问接口，以便在其上传递 IP 分组。由于这一层次未被定义，所以其具体的实现方法将随着网络类型的不同而不同。主机-网络层负责通过网络发送和接收 IP 数据报。TCP/IP 参考模型允许主机连入网络时使用多种现成的与流行的协议，例如局域网协议或其他一些协议。在 TCP/IP 的主机-网络层中，它包括各种物理网协议，例如局域网的 Ethernet、局域网的 Token Ring、分组交换网的 X.25 等。当这种物理网被用作传送 IP 数据包的通道时，我们就可以认为是这一层的内容。这体现了 TCP/IP 协议的兼容性与适应性，它也为 TCP/IP 的成功奠定了基础。

（2）互联网络层

在 TCP/IP 参考模型中，互联网络层是参考模型的第二层，它相当于 OSI 参考模型网络层的无连接网络服务。

网络互联层是整个 TCP/IP 协议栈的核心。它的功能是把分组发往目标网络或主机，源主机与目的主机可以在一个网上，也可以在不同的网上。同时，为了尽快地发送分组，可能需要沿不同的路径同时进行分组传递。因此，分组到达的顺序和发送的顺序可能不同，这就需要上层必须对分组进行排序。

网络互联层定义了分组格式和协议，即 IP 协议。

网络互联层除了需要完成路由的功能外，也可以完成将不同类型的网络（异构网）互连的任务。除此之外，网络互连层还需要完成拥塞控制的功能。

TCP/IP 参考模型中网络层协议是 IP 协议。IP 协议是一种不可靠、无连接的数据报传送服务的协议，它提供的是一种尽力而为的服务，IP 协议的协议数据单元是 IP 分组。

(3)传输层

在 TCP/IP 参考模型中，传输层负责在应用进程之间的端到端通信。传输层的主要目的是在互联网中源主机与目的主机的对等实体间建立用于会话的端到端连接。传输层的功能是使源端主机和目标端主机上的对等实体可以进行会话。

传输层定义了两种服务质量不同的协议，即：TCP 和 UDP，TCP 是一种可靠的面向连接的协议，它允许将一台主机的字节流无差错的传送到目的主机。TCP 协议同时要完成流量控制功能，协调收发双方的发送与接收速度，达到正确传输的目的。用户数据协议协议是一种不可靠的无连接协议，它主要用于不要求分组顺序到达的传输中，分组传输顺序检查与排序由应用层完成。

（4）应用层

TCP/IP 模型中，应用层是参考模型的最高层。TCP/IP 模型将 OSI 模型中的会话层和表示层的功能合并到应用层实现。目前，应用层协议主要有以下几种：

- 文件传送协议(file transfer protocol, FTP)
- 简单邮件传送协议(simple mail transfer protocol, SMTP)
- 简单网络管理协议(simple network management protocol, SNMP)
- 超文本传送协议(hyper text transfer protocol, HTTP)

有基于 TCP 协议的，如文件传输协议、超文本链接协议，也有基于 UDP 协议的协议，如简单网络管理协议等[5]。

TCP/IP 网络是分层的，每一层负责不同的通信功能。分层的概念说起来非常简单，但在实际的应用中非常的重要，在进行网络设置和排除故障时对网络层次理解得很透，将对工作有很大的帮助。例如：设置路由是网络层 IP 协议的事，要查找 MAC 地址是链路层 ARP 的事，常用的 Ping 命令由 ICMP 协议来做的[6]。

图 2-5 显示各层协议的关系：

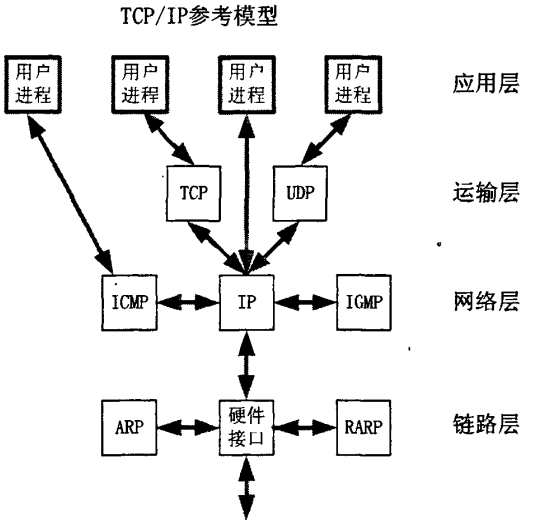

图 2-5　各层协议关系

应用程序用 TCP 传送数据时，数据被送入协议栈，逐个通过每一层直到被当作一串比特流送入网络。每层对收到的数据都要加些首部信息。TCP 传给 IP 的数据单元称作 TCP 报文段或简称 TCP 段。IP 传给网络接口层的数据单元称作 IP 数据报，如下图所示。通过以太网传输的比特流称作帧，如图 2-6 所示。数据发送自上而下，层层加码；数据接收自下而上，层层解码[7]。

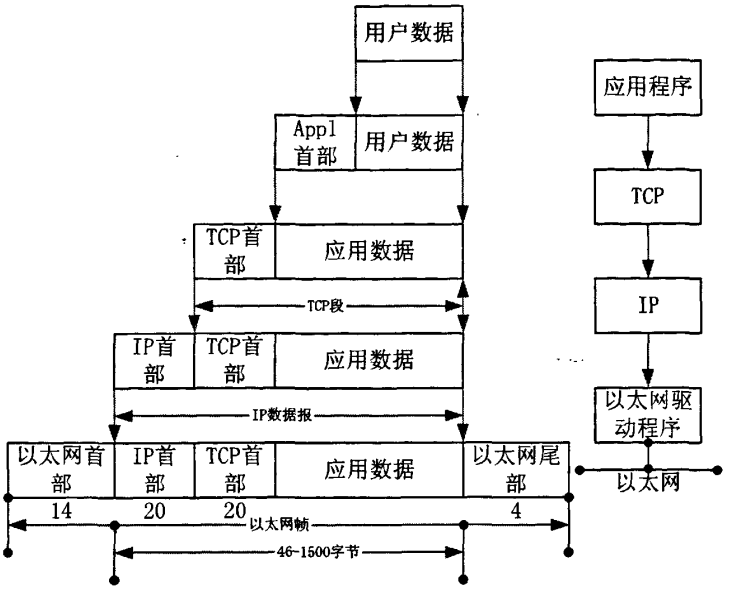

图 2-6 数据发送接收流程

为了通信，两个系统必须在各层之间传递数据、指令、地址等信息，通信的逻辑流程与真正的数据流的不同。虽然通信流程垂直通过各层次，但每一层都在逻辑上能够直接与远程计算机系统的相应层直接通信。从图 2-7 可以看出，通讯实际上是按垂直方向进行的，但在逻辑上通信是在同级进行的[8]。

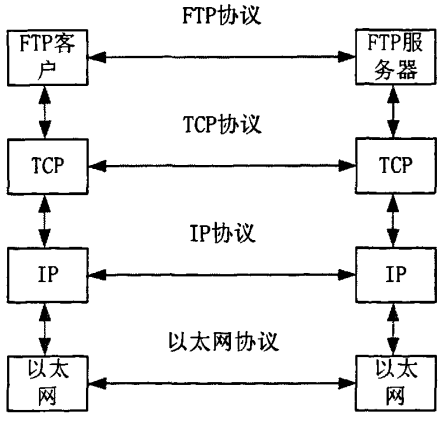

图 2-7 FTP 通信过程

第三章 主机到网络层和网络互联层

3.1 以太网和 IEEE 802 封装

在 TCP/IP 协议族中，链路层主要有三个目的：

- 为 IP 模块发送和接收 IP 数据报
- 为 ARP 模块发送 ARP 请求和接收 ARP 应答
- 为 RARP 发送 RARP 请求和接收 RARP 应答

TCP/IP 支持不同的链路层协议，如以太网、令牌环网及 RS-232 串行线路等。以太网采用称作 CSMA/CD 的媒体接入方法。IEEE802 委员会公布了一个稍有不同的标准集，其中 802.3 针对整个 CSMA/CD 网络，802.4 针对令牌总线网络，802.5 针对令牌环网络。802.2 和 802.3 定义了一个与以太网不同的帧格式[9]。图 3-1 显示了两种不同形式的封装格式。每个方框下面的数字是字节长度。

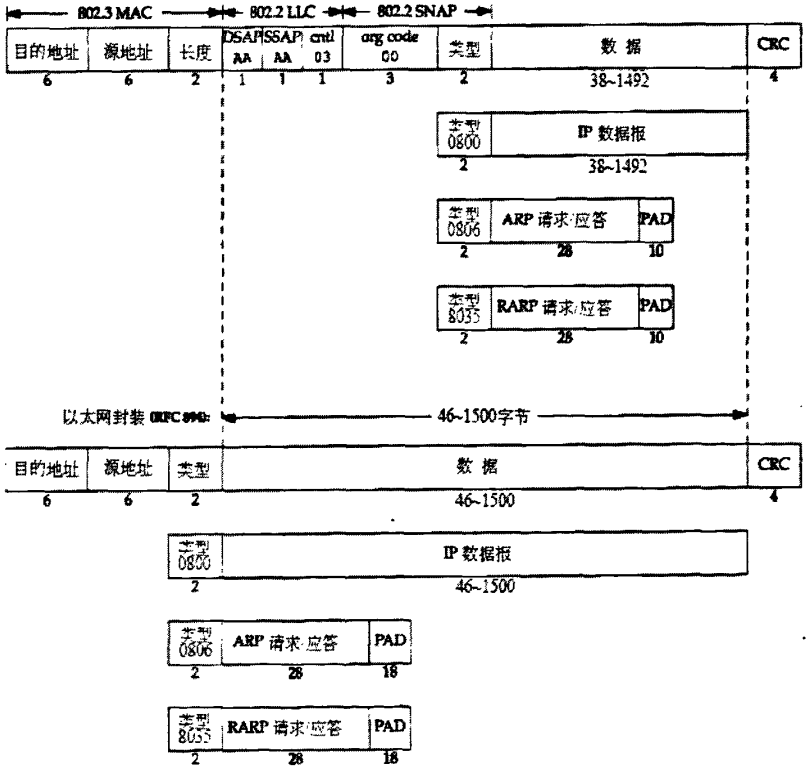

图 3-1 IEEE 802.2/802.3（RFC 1042）和以太网的封装格式（RFC 894）

两种帧格式都采用 48bit 的目的地址和源地址(802.3 允许使用 16bit 的地址，但一般是 48bit 地址)。这就是我们在本书中所称的硬件地址。ARP 和 RARP 协议对 32bit 的 IP 地址和 48bit 的硬件地址进行映射。

接下来的 2 个字节在两种帧格式中不同。在 802 标准定义的帧格式中，长度字段指它后续数据字节长度，不包括 CRC 检验码。以太网的类型字段定义了后续数据的类型。在 802 标准定义的帧格式中，类型字段则由后续的子网接入协议（Sub-network Access Protocol，SNAP）的首部给出。802 定义的有效长度值与以太网的有效类型值无一相同，这样，就可以对两种帧格式进行区分。

在以太网帧格式中，类型字段之后就是数据；而在 802 帧格式中，跟随在后面的是 3 字节的 802.2 LLC 和 5 字节的 802.2 SNAP。目的服务访问点(Destination Service Access Point, DSAP)和源服务访问点(Source Service Access Point, SSAP)的值都设为 0xaa。Ctrl 字段的值设为 3。随后的 3 个字节 org code 都置为 0。

CRC 用于后续字节差错循环冗余码检验（它也被称为 FCS 或帧检验序列）。

802.3 标准定义的帧和以太网的帧都有最小长度要求。802.3 规定数据部分必须至少为 38 字节，以太网要求最少有 46 字节。为保证这点，必须在不足的空间插入填充（pad）字节。在开始观察线路上的分组时将遇到这种最小长度的情况。

以太网是 Xerox 公司发明的基带 LAN 标准。它采用带冲突检测的载波监听多路访问协议（CSMA / CD），速率为 10Mbps，传输介质为同轴电缆。以太网是在 20 世纪 70 年代为解决网络中零散的和偶然的堵塞而开发的，而 IEEE802. 3 标准是在最初的以太网技术基础上于 1980 年开发成功的。现在，以太网一词泛指所有采用 CSMA / CD 协议的局域网。以太网 2．0 版由数字设备公司、Intel 公司和 Xerox 公司联合开发，它与 IEEE802．3 兼容。

以太网和 IEEE802．3 通常由接口卡（网卡）或主电路板上的电路实现。以太网电缆协议规定用收发器将电缆连到网络物理设备上。收发器执行物理层的大部分功能，其中包括冲突检测及收发器电缆将收发器连接到工作站上。

IEEE802．3 提供了多种电缆规范，10Base5 就是其中的一种，它与以太网最为接近。在这一规范中，连接电缆称作连接单元接口（AUI），网络连接设备称为介质访问单元（MAU）而不再是收发器。

在基于广播的以太网中，所有的工作站都可以收到发送到网上的信息帧。每个工作站都要确认该信息帧是不是发送给自己的，一旦确认是发给自己的，就将它发送到高一层的协议层。在采用 CSMA / CD 传输介质访问的以太网中，任何一个 CSMA / CDLAN 工作站在任何一时刻都可以访问网络。发送数据前，工作站要侦听网络是否堵塞，只有检测到网络空闲时，工作站才能发送数据。在基于竞争的以太网中，只要网络空闲，任一工作站均可发送数据。当两个工作站发现

网络空闲而同时发出数据时，就发生冲突。这时，两个传送操作都遭到破坏，工作站必须在一定时间后重发，何时重发由延时算法决定[10]。

尽管以太网与 IEEE802．3 标准有很多相似之处，但也存在一定的差别。以太网提供的服务对应于 OSI 参考模型的第一层和第二层，而 IEEE802．3 提供的服务对应于 OSI 参考模型的第一层和第二层的信道访问部分（即第二层的一部分）。IEEE802．3 没有定义逻辑链路控制协议，但定义了几个不同物理层，而以太网只定义了一个。

IEEE802．3 的每个物理层协议都可以从三方面说明其特征，这三方面分别是 LAN 的速度、信号传输方式和物理介质类型。

RFC 893[Leffler and Karels 1984]描述了另一种用于以太网的封装格式，称作尾部封装（trailer encapsulation）。这是一个早期 BSD 系统在 DEC VAX 机上运行时的试验格式，它通过调整 IP 数据报中字段的次序来提高性能。在以太网数据帧中，开始的那部分是变长的字段（IP 首部和 TCP 首部）。把它们移到尾部（在 CRC 之前），这样当把数据复制到内核时，就可以把数据帧中的数据部分映射到一个硬件页面，节省内存到内存的复制过程。TCP 数据报的长度是 512 字节的整数倍，正好可以用内核中的页表来处理。两台主机通过协商使用 ARP 扩展协议对数据帧进行尾部封装。这些数据帧需定义不同的以太网帧类型值。现在，尾部封装已遭到反对，因此我们不对它举任何例子[11]。

3.2 IP 协议

IP 是 TCP/IP 协议族中最为核心的协议。所有的 TCP、UDP、ICMP 及 IGMP 数据都以 IP 数据报格式传输。有个形象的比喻 IP 协议就像运货的卡车，将一车车的货物运向目的地。主要的货物就是 TCP 或 UDP 分配给它的。IP 提供不可靠、无连接的数据报传送服务，不可靠（unreliable）的意思是它不能保证 IP 数据报能成功地到达目的地。IP 仅提供最好的传输服务但不保证 IP 数据报能成功地到达目的地。如果发生某种错误时，如某个路由器暂时用完了缓冲区，IP 有一个简单的错误处理算法：丢弃该数据报，然后发送 ICMP 消息报给信源端。任何要求的可靠性必须由上层来提供（如 TCP）。无连接（connectionless）这个术语的意思是 IP 并不维护任何关于后续数据报的状态信息。

IP 数据报格式如图 3-2 所示。普通 IP 首部长为 20 个字节，除非含选项字段。

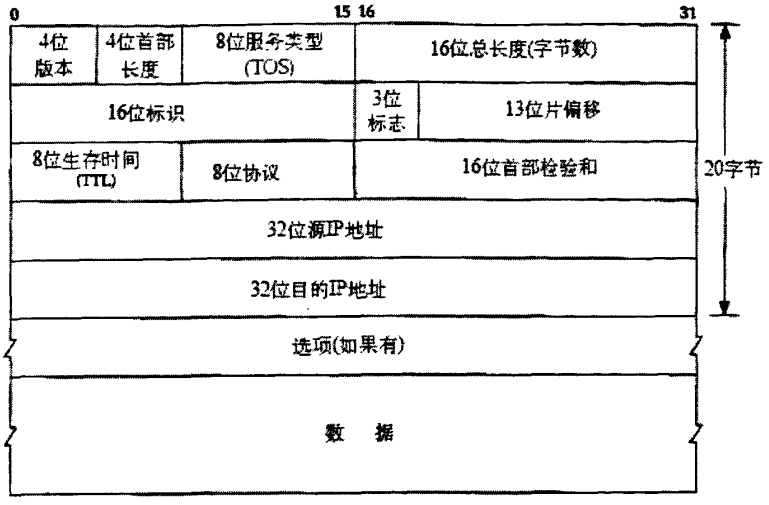

图 3-2 IP 数据报格式及首部中的各字段

分析图 3-2 中的首部。最高位在左边，记为 0bit；最低位在右边，记为 31bit。4 个字节的 32bit 值以下面的次序传输：首先是 0～7bit，其次 8～15bit，然后 16～23bit，最后是 24~31bit。这种传输次序称作 big endian 字节序。由于 TCP/IP 首部中所有的二进制整数在网络中传输时都要求以这种次序，因此它又称作网络字节序。以其他形式存储二进制整数的机器，如 little endian 格式，则必须在传输数据之前把首部转换成网络字节序。

4 位版本：版本号（Version）,表示目前的协议版本号，长度 4 比特。标识目前采用的 IP 协议的版本号。一般的值为 0100（IPv4），IPv6 的值（0110）。

4 位首部长度：指的是 IP 包头长度（Header Length）,头部的是长度，它的单位是 32 位(4 个字节)，长度 4 比特,数值为 5 表示 IP 头部长度为 20 字节,这个字段的作用是为了描述 IP 包头的长度，因为在 IP 包头中有变长的可选部分。IP 包头最小长度为 20 字节，由于变长的可选部分最大长度可能会变成 24 字节。

8 位服务类型(TOS)：这个 8 位字段由 3 位的优先权子字段，现在已经被忽略，4 位的 TOS 子字段以及 1 位的未用字段（现在为 0）构成。4 位的 TOS 子字段包含：最小延时、最大吞吐量、最高可靠性以及最小费用构成，这四个 1 位最多只能有一个为 1，都为 0，表示是一般服务。这个子段可以拆分成两个部分：Precedence 和 TOS。TOS 目前不太使用。而 Precedence 则用于 QOS 应用。（TOS 字段的详细描述 RFC 1340 1349）

16 位总长度(字节数)：总长度字段是指整个 IP 数据报的长度，以字节为单位。这个数据报只是传送的控制信息，还没有传送真正的数据，所以目前看到的总长度就是报头的长度。IP 包最大长度 65535 字节。

16 位标识：该字段唯一标识主机发送的数据报。每发一份报文值就加 1，该字段和 Flags 和 Fragment Offset 字段联合使用，对大的上层数据包进行分段操作。

标记（Flags）：长度 3 比特。该字段第一位不使用。第二位是 DF 位，DF 位设为 1 时表明路由器不能对该上层数据包分段。如果一个上层数据包无法在不分段的情况下进行转发，则路由器会丢弃该上层数据包并返回一个错误信息。第三位是 MF 位，当路由器对一个上层数据包分段，则路由器会在除了最后一个分段的 IP 包的包头中将 MF 位设为 1。

13 位片偏移,又称分段序号（Fragment Offset）,长度 13 比特。该字段对包含分段的上层数据包的 IP 包赋予序号。由于 IP 包在网络上传送的时候不一定能按顺序到达，这个字段保证目标路由器在接受到 IP 包之后能够还原分段的上层数据包。到某个包含分段的上层数据包的 IP 包在传送是丢失，则整个一系列包含分段的上层数据包的 IP 包都会被要求重传。

生存时间（TTL）：TTL（time-to-live）,长度 8 比特。生存时间字段设置了数据报可以经过的最多路由器数。它指定了数据报的生存时间。可根据 TTL 值判断服务器是什么系统和经过的路由器。ttl 的初始值由源主机设置，当 IP 包进行传送时，先会对该字段赋予某个特定的值。当 IP 包经过每一个沿途的路由器的时候，每个沿途的路由器会将 IP 包的 TTL 值减少 1。如果 TTL 减少为 0，则该 IP 包会被丢弃。这个字段可以防止由于故障而导致 IP 包在网络中不停被转发。

8 位协议：表示协议类型，6 表示传输层是 TCP 协议。长度 8 比特。为 1 时代表 ICMP 协议，为 6 时代表 TCP 协议，为 17 时代表 UDP 协议。

16 位首部检验和：由于 IP 包头是变长的，所以提供一个头部校验来保证 IP 包头中信息的正确性。当收到一份 IP 数据报后，同样对首部中每个 16 位进行二进制反码的求和。由于接收方在计算过程中包含了发送方存在首部中的检验和，因此，如果首部在传输过程中没有发生任何差错，那么接收方计算的结果应该为全 1。如果结果不是全 1，即检验和错误，那么 IP 就丢弃收到的数据报。但是不生成差错报文，由上层去发现丢失的数据报并进行重传。

32 位源 IP 地址和 32 位目的 IP 地址：这是 IP 协议核心的部分，标识了起源和目标地址。32 位的 IP 地址由一个网络 ID 和一个主机 ID 组成。若两台主机网络地址是相同的,则在一个网段内，这样数据在传送过程中可直接到达。

可选项（Options）：这是可变长的字段。该字段由起源设备根据需要改写。

路由记录（Record route）：IP 包离开路由器时记录路由器出站接口 IP 地址。

时间戳（Timestamps）：当 IP 包离开每个路由器的时候记录时间。

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（以太网或令牌环网），

那么 IP 数据报就直接送到目的主机上。否则，主机把数据报发往一默认的路由器上，由路由器来转发该数据报。大多数的主机都是采用这种简单机制。

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，或者从一个网络接口接收数据报（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，它都要对该表搜索一次。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 地址之一或者 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 层被设置为路由器的功能，那么就对数据报进行转发（也就是说，像下面对待发出的数据报一样处理）；否则数据报被丢弃。

3.3 地址解析协议

数据链路如以太网或令牌环网都有自己的寻址机制（常常为 48bit 地址），这是使用数据链路的任何网络层都必须遵从的。一个网络如以太网可以同时被不同的网络层使用。例如，一组使用 TCP/IP 协议的主机和另一组使用某种 PC 网络软件的主机可以共享相同的电缆。

当主机把以太网数据帧发送到位于同一局域网上的另一主机时，根据 48bit 的以太网地址确定目的接口的。驱动程序从不检查 IP 数据报中的目的 IP 地址。

地址解析为这两种不同的地址形式提供映射：32bit 的 IP 地址和数据链路层使用的任何类型的地址。

ARP 为 IP 地址到对应的硬件地址之间提供动态映射。我们之所以用动态这个词是因为这个过程是自动完成的，一般应用程序用户或系统管理员不必关心。

RARP 是被那些没有磁盘驱动器的系统使用(一般是无盘工作站或 X 终端)，它需要系统管理员进行手工设置。

在 ARP 背后有一个基本概念，那就是网络接口有一个硬件地址（一个 48bit 的值，标识不同的以太网或令牌环网络接口）。在硬件层次上进行的数据帧交换必须有正确的接口地址。但是，TCP/IP 有自己的地址：32bit 的 IP 地址。知道主机的 IP 地址并不能让内核发送一帧数据给主机。内核（如以太网驱动程序）必须知道目的端的硬件地址才能发送数据。ARP 的功能是在 32bit 的 IP 地址和采用不同网络技术的硬件地址之间提供动态映射。

在以太网上解析 IP 地址时，ARP 请求和应答分组的格式如图 3-3 所示：ARP 可以用于其他类型的网络，可以解析 IP 地址以外的地址。紧跟着帧类型字段的前四个字段指定了最后四个字段的类型和长度。

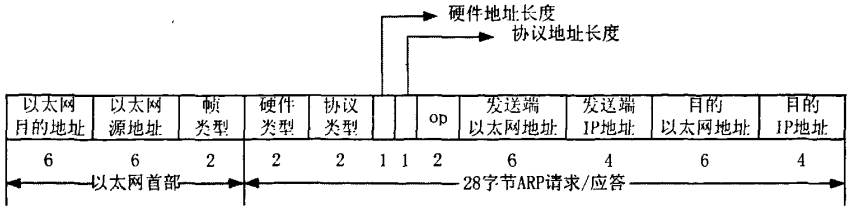

图 3-3 用于以太网的 ARP 请求或应答分组格式

以太网报头中的前两个字段是以太网的源地址和目的地址。目的地址为全 1 的特殊地址是广播地址。电缆上的所有以太网接口都要接收广播的数据帧。

两个字节长的以太网帧类型表示后面数据的类型。对于 ARP 请求或应答来说，该字段的值为 0x0806。数据报的以太网数据帧中的类型字段的值相同。

硬件类型字段表示硬件地址的类型。它的值为 1 即表示以太网地址。

协议类型字段表示要映射的协议地址类型。值为 0800 即表示 IP 地址。它的值与包含 IP 数据报的以太网数据帧中的类型字段的值相同，这是有意设计的。

接下来的两个 1 字节的字段，硬件地址长度和协议地址长度分别指出硬件地址和协议地址的长度，以字节为单位。对于以太网上 IP 地址的 ARP 请求或应答来说，它们的值分别为 6 和 4。

Op 即操作（operation），操作字段指出四种操作类型，它们是 ARP 请求（值为 1）、ARP 应答（值为 2）、RARP 请求（值为 3）和 RARP 应答（值为 4）。这个字段必需的，因为 ARP 请求和 ARP 应答的帧类型字段值是相同的。

接下来的四个字段是发送端的硬件地址、发送端的 IP 地址、目的端的硬件地址和目的端 IP 地址。注意，这里有一些重复信息：在以太网的数据帧报头中和 ARP 请求数据帧中都有发送端的硬件地址。对于一个 ARP 请求来说，除目的端硬件地址外的所有其他的字段都有填充值。当系统收到一份目的端为本机的 ARP 请求报文后，它就把硬件地址填进去，然后用两个目的端地址分别替换两个发送端地址，并把操作字段置为 2，最后把它发送回去。

ARP 协议是一种解析协议，本来主机是完全不知道这个 IP 对应的是哪个主机的哪个接口，当主机要发送一个 IP 包的时候，会首先查一下自己的 ARP 高速缓存（就是一个 IP-MAC 地址对应表缓存），如果查询的 IP—MAC 值对不存在，那么主机就向网络发送一个 ARP 协议广播包，这个广播包里面就有待查询的 IP 地址，而直接收到这份广播的包的所有主机都会查询自己的 IP 地址，如果收到

广播包的某一个主机发现自己符合条件，那么就准备好一个包含自己的 MAC 地址的 ARP 包传送给发送 ARP 广播的主机，而广播主机拿到 ARP 包后会更新自己的 ARP 缓存（就是存放 IP-MAC 对应表的地方）。发送广播的主机就会用新的 ARP 缓存数据准备好数据链路层的的数据包发送工作。

RARP 分组格式与 ARP 分组基本一致。它们之间的差别是 RARP 请求或应答的帧类型代码为 0x8035，RARP 请求的操作代码为 3，应答操作代码为 4。RARP 请求在网络上进行广播，它在分组中标明发送端硬件地址，请求相应 IP 地址的响应。RARP 请求以广播方式传送，而 RARP 应答一般是单播(unicast)传送的。

3.4 网际控制报文协议

ICMP 全称 Internet Control Message Protocol（网际控制信息协议）。为了处理错误，TCP/IP 设计了 ICMP 协议，当某个网关发现传输错误时，立即向信源主机发送 ICMP 报文，报告出错信息，让信源主机采取相应处理措施，它是一种差错和控制报文协议。ICMP 经常被认为是 IP 层的一个组成部分。它传递差错报文以及其他需要注意的信息。ICMP 报文通常被 IP 层或更高层协议（TCP 或 UDP）使用。一些 ICMP 报文把差错报文返回给用户进程。

ICMP 报文包含在 IP 数据报中，属于 IP 的一个用户，IP 头部就在 ICMP 报文的前面，所以一个 ICMP 报文包括 IP 头部、ICMP 头部和 ICMP 报文，IP 头部的 Protocol 值为 1 就说明这是一个 ICMP 报文，ICMP 头部中的类型（Type）域用于说明 ICMP 报文的作用及格式，此外还有一个代码（Code）域用于详细说明某种 ICMP 报文的类型，所有数据都在 ICMP 头部后面。

ICMP 报文是在 IP 数据报内部被传输的，如图 3-4 所示。

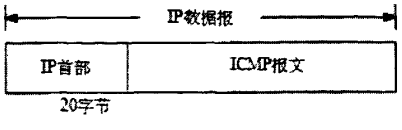

图 3-4　ICMP 报文封装于 IP 数据报

ICMP 报文的格式如图 3-5 所示。所有报文的前 4 个字节都是一样的，但是剩下的其他字节则互不相同。类型字段可以有 15 个不同的值，以描述特定类型的 ICMP 报文。某些 ICMP 报文还使用代码字段的值来进一步描述不同的条件。检验和字段覆盖整个 ICMP 报文。ICMP 的检验和是必需的。

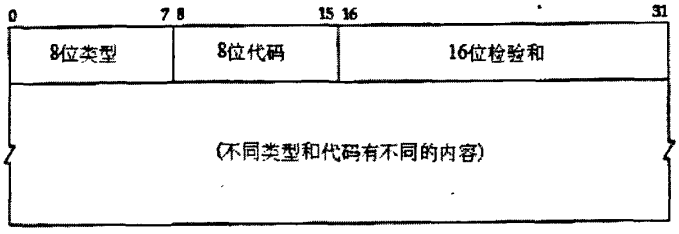

图 3-5 ICMP 报文的格式

RFC 定义了 13 种 ICMP 报文格式，具体如下：

0 响应应答（ECHO-REPLY）

3 不可到达

4 源抑制

5 重定向

8 响应请求（ECHO-REQUEST）

11 超时

12 参数失灵

13 时间戳请求

14 时间戳应答

15 信息请求（*已作废）

16 信息应答（*已作废）

17 地址掩码请求

18 地址掩码应答

其中代码为 15、16 的信息报文已经作废。

下面是几种常见的 ICMP 报文：

1.响应请求

我们日常使用最多的 ping，就是响应请求（Type=8）和应答（Type=0），一台主机向一个节点发送一个 Type=8 的 ICMP 报文，如果途中没有异常（例如被路由器丢弃、目标不回应 ICMP 或传输失败），则目标返回 Type=0 的 ICMP 报文，说明这台主机存在。

2.目标不可到达、源抑制和超时报文

这三种报文的格式是一样的，目标不可到达报文（Type=3）在路由器或主机不能传递数据报时使用，常见的不可到达类型还有网络不可到达（Code=0）、主机不可到达（Code=1）、协议不可到达（Code=2）等。源抑制则充当一个控制流量的角色，它通知主机减少数据报流量，由于 ICMP 没有恢复传输的报文，所以只要停止该报文，主机就会逐渐恢复传输速率。最后，无连接方式网络的问

题就是数据报会丢失，或者长时间在网络游荡而找不到目标，或者拥塞导致主机在规定时间内无法重组数据报分段，这时就要触发 ICMP 超时报文的产生。超时报文的代码域有两种取值：Code=0 表示传输超时，Code=1 表示重组分段超时。

ICMP 不可达报文的一般格式如图 3-6 所示:

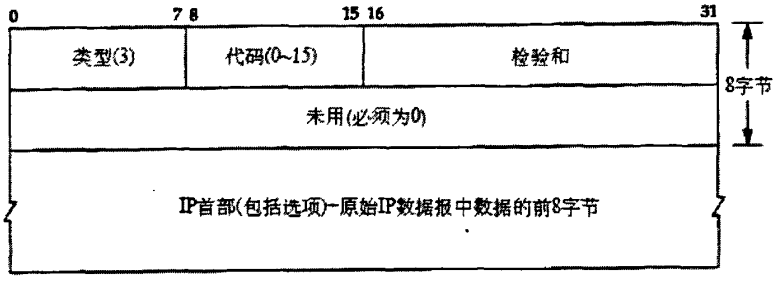

图 3-6 ICMP 不可达报文

3.时间戳

时间戳请求报文（Type=13）和时间戳应答报文（Type=14）用于测试两台主机之间数据报来回一次的传输时间。

应用层的 PING(Packet Internet Groper)命令用来测试两个主机之间的连通性，PING 使用了 ICMP 回送请求与回送回答报文，属于 ICMP 询问报文，它是应用层直接使用网络层 ICMP 的一个特例，它没有通过运输层的 TCP 或 UDP。

对于携带 ICMP 差错报文的数据报，不再产生 ICMP 差错报文；对于分片的数据报，如果不是第一个分片，则不产生 ICMP 差错报文；对于特殊地址如 127.0.0.0，0.0.0.0 或多播地址不产生 ICMP 差错报文。所有的差错报文都包括数据部分，而这数据部分包括原始数据报的首部加上数据报中的前 8 个字节。加上原始数据报中的首部的目的是为了向接收差错报文的原始信源给出关于数据报本身的信息，要包括数据的前 8 个字节是国为这前 8 个字节提供了关于端口号（UDP 和 TCP）和序号（TCP）的信息。

第四章 主机对主机层

4.1 用户数据报协议

UDP 协议是英文 User Datagram Protocol 的缩写，即用户数据报协议，主要用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用 UDP 协议。UDP 协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但是即使是在今天，UDP 仍然不失为一项非常实用和可行的网络传输层协议。UDP 协议的主要作用是将网络数据流量压缩成数据报的形式。一个典型的数据报就是一个二进制数据的传输单位。每一个数据报的前 8 个字节用来包含报头信息，剩余字节则用来包含具体的传输数据。

用户数据报协议（UDP）是一种无连接的传输层协议，提供面向操作的简单不可靠信息传送服务。UDP 协议直接工作于 IP 协议的顶层。UDP 协议端口不同于多路应用程序，其运行是从一个单个设备到另一个单个设备。

大多数网络应用程序都是在相同的机器上运行。计算机上必须能确保目的地的正确软件应用程序从源地址处获得数据包，以及源计算机上的正确应用程序的回复获得选择路经。这一过程是通过使用 UDP 的端口号完成的。例如，如果一个工作站希望在站 128.1.123.1 上使用域名系统，它就得对欲连接的站 128.1.123.1 的包进行寻址操作并在 UDP 头插入目标端口号 53。源端口号确定被请求域名服务的本地机的应用程序，同时需要对所有由目的站生成的响应包进行寻址。与 TCP 不同，UDP 并不提供数据传送的可靠机制、流控制以及错误恢复功能等。由于 UDP 比较简单，UDP 头包含很少的字节，比 TCP 消耗少。

UDP 数据报封装成一份 IP 数据报的格式，如图 4-1 所示:

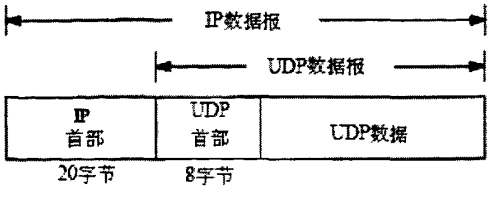

图 4-1 UDP 封装

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。

UDP 信息包由 UDP 标题和数据组成。UDP 的标题结构由 5 个域组成：源端端口(Source Port)、目的地端口(Destination Port)、用户数据包的长度(Length)和检查和（Checksum)。其中，前 4 个域组成 UDP 标题(UDP header)，每个域由 4 个字节组成；检查和域占据 2 个字节，它用来检测传输过程中是否出现了错误；用户数据包的长度包括所有 5 个域的字节数。UDP 首部的各字段如图 4-2 所示：

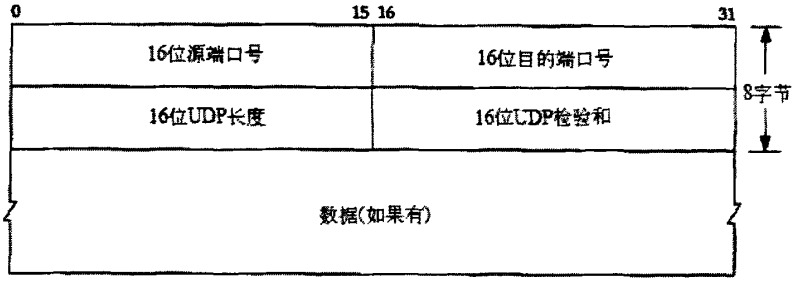

图 4-2 UDP 首部

源端口是可选域，当其有意义时，它指的是发送进程的端口，这也就假定了在没有其它信息的情况下，返回信息应该向什么地方发送。如果不使用它，则在此域中填 0。目的端口在有特定的目的网络地址时有意义。长度指的是此用户数据报长度的八进制表示。校验码有 16 位，是对 IP 头，UDP 头和数据中信息包头的数位取反求和再取反得到的。

端口号表示发送进程和接收进程。TCP 和 UDP 用目的端口号来分用来自 IP 层的数据的过程。由于 IP 层已经把 IP 数据报分配给 TCP 或 UDP，因此 TCP 端口号由 TCP 来查看，而 UDP 端口号由 UDP 来查看。TCP 端口号与 UDP 端口号是相互独立的。

尽管相互独立，如果 TCP 和 UDP 同时提供某种知名服务，两个协议通常选择相同的端口号。这纯粹是为了使用方便，而不是协议本身的要求。

UDP 长度字段指的是 UDP 首部和 UDP 数据的字节长度。该字段的最小值为 8 字节。这个 UDP 长度是有冗余的。IP 数据报长度指的是数据报全长，UDP 数据报长度是全长减去 IP 首部的长度。

UDP 适用于不需要 TCP 可靠机制的情形，如：高层协议或应用程序提供错误和流控制的情况。 UDP 是传输层协议，应用于个别应用层协议，包括网络文件系统（NFS)、简单网络管理协议（SNMP)、域名系统（DNS）以及简单文件传输系统（TFTP)。

UDP 和 TCP 首部都包含一个 12 字节的伪首部，包含了 IP 首部和自身的一些字段，主要是为了计算检验和而设置的。伪首部是不占实际空间的。伪首部包含 IP 首部的一些字段，目的是让 UDP 两次检查数据是否已经到达目的地，以及 IP 层是否正确地传输了数据。

UDP 检验和覆盖 UDP 首部和 UDP 数据。回想 IP 首部的检验和，它只覆盖 IP 的首部—并不覆盖 IP 数据报中的任何数据。UDP 和 TCP 在首部中都有覆盖它们首部和数据的检验和。UDP 的检验和是可选的，而 TCP 的检验和是必需的。

UDP 检验和的基本计算方法与 IP 首部检验和计算方法相类似（16bit 字的二进制反码和），但是它们之间存在不同的地方。首先，UDP 数据报的长度可以为奇数字节，但是检验和算法是把若干个 16bit 字相加。解决方法是必要时在最后增加填充字节 0，这只是为了检验和的计算。

UDP 数据报中的伪首部格式如图 4-3 所示：

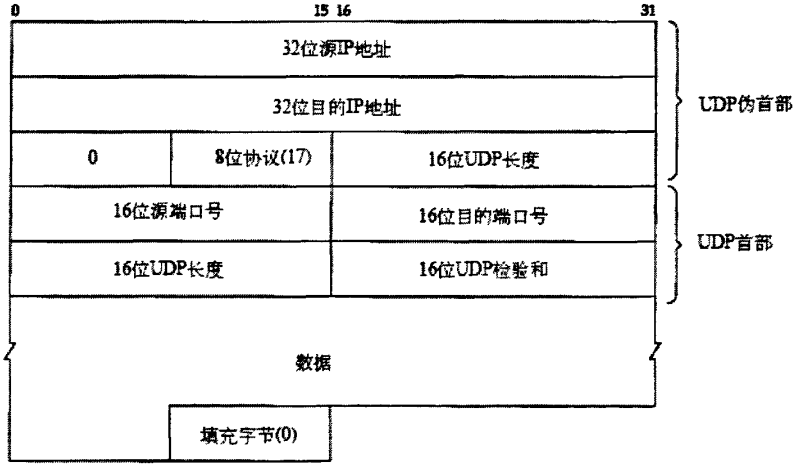

图 4-3　UDP 数据报中的伪首部

如果发送端没有计算检验和而接收端检测到检验和有差错，那么 UDP 数据报就要被悄悄地丢弃。UDP 检验和是一个端到端的检验和。它由发送端计算，然后由接收端验证。其目的是为了发现 UDP 首部和数据在发送端到接收端之间发生的任何改动。

检查和的详细计算可在 RFC 1071 中找到，现举一例说明使用检查和检测错误的道理。例如，假设从源端 A 要发送下列 3 个 16 位的二进制数：word1，word2 和 word3 到终端 B，检查和计算如下：

word1　　0110011001100110

word2	0101010101010101
word3	0000111100001111
sum=word1+word2+word3	1100101011001010
检查和(sum 的反码)	0011010100110101

从发送端发出的 4 个(word1，2，3 以及检查和)16 位二进制数之和为1111111111111111，如果接收端收到的这 4 个 16 位二进制数之和也是全“1”，就认为传输过程中没有出差错。

许多链路层协议都提供错误检查，包括流行的以太网协议，UDP 也要提供检查和,其原因是链路层以下的协议在源端和终端之间的某些通道可能不提供错误检测。虽然 UDP 提供有错误检测，但检测到错误时，UDP 不做错误校正，只是简单地把损坏的消息段扔掉，或者给应用程序提供警告信息。

下面介绍 UDP 协议的几个特性:

(1) UDP 是一个无连接协议，传输数据之前源端和终端不建立连接，当它想传送时就简单地去抓取来自应用程序的数据，并尽可能快地把它扔到网络上。在发送端，UDP 传送数据的速度仅仅是受应用程序生成数据的速度、计算机的能力和传输带宽的限制；在接收端，UDP 把每个消息段放在队列中，应用程序每次从队列中读一个消息段。

(2) 由于传输数据不建立连接，因此也就不需要维护连接状态，包括收发状态等，因此一台服务机可同时向多个客户机传输相同的消息。

(3) UDP 信息包的标题很短，只有 8 个字节，相对于 TCP 的 20 个字节信息包的额外开销很小。

(4) 吞吐量不受拥挤控制算法的调节，只受应用软件生成数据的速率、传输带宽、源端和终端主机性能的限制。

虽然 UDP 是一个不可靠的协议，但它是分发信息的一个理想协议。例如，在屏幕上报告股票市场、在屏幕上显示航空信息等等。UDP 也用在路由信息协议 RIP(Routing Information Protocol)中修改路由表。在这些应用场合下，如果有一个消息丢失，在几秒之后另一个新的消息就会替换它。UDP 广泛用在多媒体应用中，例如，Progressive Networks 公司开发的 RealAudio 软件，它是在因特网上把预先录制的或者现场音乐实时传送给客户机的一种软件，该软件使用的 RealAudio audio-on-demand protocol 协议就是运行在 UDP 之上的协议，大多数因特网电话软件产品也都运行在 UDP 之上。

4.2 传输控制协议

TCP 是面向连接的端到端的可靠协议。它支持多种网络应用程序。TCP 对下层服务没有多少要求，它假定下层只能提供不可靠的数据报服务，它可以在多种硬件构成的网络上运行。TCP 下层是 IP 协议，TCP 可以根据 IP 协议提供的服务传送大小不定的数据，IP 协议负责对数据进行分段，重组，在多种网络中传送。TCP 的上面就是应用程序，下面是 IP 协议，上层接口包括一系列类似于操作系统中断的调用。对于上层应用程序来说，TCP 应该能够异步传送数据。下层接口我们假定为 IP 协议接口。为了在并不可靠的网络上实现面向连接的可靠的传送数据，TCP 必须解决可靠性，流量控制的问题，必须能够为上层应用程序提供多个接口，同时为多个应用程序提供数据，同时 TCP 必须解决连接问题，这样 TCP 才能称得上是面向连接的，最后，TCP 也必须能够解决通信安全性的问题。在实现 TCP 的主机上，TCP 可以被看成是一个模块，和文件系统区别不大，TCP 也可以调用一些操作系统的功能，TCP 不直接和网络打交道，控制网络的任务由专门的设备驱动模块完成。TCP 只是调用 IP 接口，IP 向 TCP 提供所有 TCP 需要的服务。

TCP 数据被封装在一个 IP 数据报中，如图 4-4 所示：

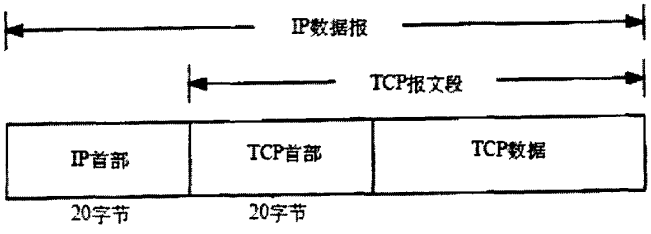

图 4-4 TCP 数据在 IP 数据报中的封装

TCP 首部的数据格式，见图 4-5。如果不计任选字段，它通常是 20 个字节。

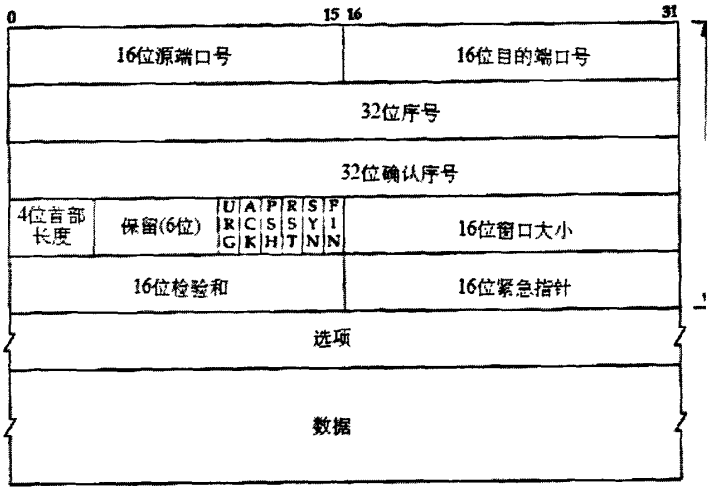

图 4-5 TCP 包首部

下面详细介绍一下 TCP 协议头信息:

端口号：常说 FTP 占 21 端口、HTTP 占 80 端口、TELNET 占 23 端口等，这里指的端口就是 TCP 或 UDP 的端口，端口就像通道两端的门一样，当两机进行通讯时门必须是打开的。源端口和目的端口各占 16 位，2 的 16 次方等于 65536，这就是每台电脑与其它电脑联系所能开的“门”。一般作为服务一方每项服务的端口号是固定的。但是数据传送时用另一端口，客户端与服务器联系时随机开一个大于 1024 的端口 4。你的电脑中了木马也会开一个服务端口。观察端口非常重要，不但能看出本机提供的正常服务，还能看出不正常的连接。Windows 察看端口的命令时 netstat。源端和目的端的端口号，用于寻找发端和收端应用进程。这两个值加上 IP 首部中的源端 IP 地址和目的端 IP 地址唯一确定一个 TCP 连接。一个 IP 地址和一个端口号也称为一个插口（socket）。这个术语出现在最早的 TCP 规范中，后来它也作为表示伯克利版的编程接口。插口对（socket pair）(包含客户 IP 地址、客户端口号、服务器 IP 地址和服务器端口号的四元组)可唯一确定互联网络中每个 TCP 连接的双方。

32 位序号：也称为顺序号（Sequence Number），简写为 SEQ，从上面三次握手的分析可以看出，当一方要与另一方联系时就发送一个初始序号给对方，意思是：“让我们建立联系吧？”，服务方收到后要发个独立的序号给发送方，意思是“消息收到，数据流将以这个数开始。”由此可看出，TCP 连接完全是双向的，即双方的数据流可同时传输。在传输过程中双方数据是独立的，因此每个 TCP 连接必须有两个顺序号分别对应不同方向的数据流。32 位序号用来标识从 TCP 发端向 TCP 收端发送的数据字节流，它表示在这个报文段中的的第一个数

据字节。如果将字节流看作在两个应用程序间的单向流动，则 TCP 用序号对每个字节进行计数。序号是 32bit 的无符号数，序号到达 $2^{32}-1$ 后又从 0 开始。序号字段包含由这个主机选择的该连接的初始序号 ISN（Initial Sequence Number）。该主机要发送数据的第一个字节序号为这个 ISN 加 1，因为 SYN 标志消耗了一个序号。TCP 为应用层提供全双工服务。这意味数据能在两个方向上独立地进行传输。因此，连接的每一端必须保持每个方向上的传输数据序号。

32 位确认序号：也称为应答号（Acknowledgment Number），简写为 ACK。在握手阶段，确认序号将发送方的序号加 1 作为回答，在数据传输阶段，确认序号将发送方的序号加发送的数据大小作为回答，表示确实收到这些数据。确认序号包含发送确认的一端所期望收到的下一个序号。因此，确认序号应当是上次已成功收到数据字节序号加 1。TCP 可以表述为一个没有选择确认或否认的滑动窗口协议。我们说 TCP 缺少选择确认是因为 TCP 首部中的确认序号表示发方已成功收到字节，但还不包含确认序号所指的字节。当前还无法对数据流中选定的部分进行确认。例如，如果 1～1024 字节已经成功收到，下一报文段中包含序号从 2049～3072 的字节，收端并不能确认这个新的报文段。它所能做的就是发回一个确认序号为 1025 的 ACK。它也无法对一个报文段进行否认。例如，如果收到包含 1025～2048 字节的报文段，但它的检验和错，TCP 接收端所能做的就是发回一个确认序号为 1025 的 ACK。

4 位首部长度：这个字段占 4 位，它的单位时 32 位（4 个字节）。TCP 的头长度最长可为 60 字节（二进制 1111 换算为十进制为 15，15*4 字节=60 字节）。4 位首部长度给出首部中 32bit 字的数目。需要这个值是因为任选字段的长度是可变的。这个字段占 4bit，因此 TCP 最多有 60 字节的首部。然而，没有任选字段，正常的长度是 20 字节。

在 TCP 首部中有 6 个标志比特。它们中的多个可同时被设置为 1。

URG　紧急指针（urgent pointer）有效。告诉接收 TCP 模块紧要指针域指着紧要数据。

ACK　确认序号有效。只有 ACK 标志为 1 时确认序号字段才有效。为 0 的时候表示数据段不包含确认信息，确认号被忽略。发送 ACK 无需任何代价，因为 32bit 的确认序号字段和 ACK 标志一样，总是 TCP 首部的一部分。因此，我们看到一旦一个连接建立起来，这个字段总是被设置，ACK 标志也总是被设置为 1。TCP 协议使用重新发送与正向 ACK 来保证数据传输的可靠性。

PSH　接收方应该尽快将这个报文段交给应用层；

RST　置 1 时重建连接。如果接收到 RST 位时候，通常发生了某些错误；

SYN　同步序号用来发起一个连接。当建立一个新的连接时，SYN 标志变 1；

FIN 发端完成发送任务。FIN 置 1 时表示发端完成发送任务。用来释放连接，表明发送方已经没有数据发送了。

TCP 协议在能够发送数据之前就建立起了连接。要实现这个连接，启动 TCP 连接的那一方首先将发送一个 SYN 数据包。这只是一个不包含数据的数据包，然后，打开 SYN 标记。如果另一方同时在它收到 SYN 标记的端口通话，它将发回一个 SYN+ACK：SYN 和 ACK 标志位都被打开，并将 ACK 编号字段设定为刚收到的那个数据包的顺序号字段的值。接下来，连接发起方为了表示收到了这个 SYN+ACK 信息，会向发送方发送一个最终的确认信息（ACK 包）。这种 SYN、SYN+ACK、ACK 的步骤被称为 TCP 连接建立时的“三次握手”。在这之后，连接就建立起来了。这个连接将一直保持活动状态，直到超时或者任何一方发出一个 FIN(结束)信号。任何一方都可以关闭一个 TCP 连接，要求双方发送一个 FIN 信号关闭自己的通讯频道。一方可以在另一方之前关闭，或者双方同时关闭 TCP 连接。因此，当一方发送一个 FIN 信号时，另一方可发送“FIN+ACK”，开始关闭自己一方的通信并且确认收到了第一个 FIN 信号。发送第一个 FIN 信号的人接下来再发送一个“FIN+ACK”信息，确认收到第二个 FIN 信号。另一方就知道这个连接已经关闭了，并且关闭了自己的连接。发送第一个 FIN 的人没有办法收到最后一个 ACK 信号的确认信息。这时它会进入“TIME_WAIT”(等待时间)状态并启动一个定时器，防止另一方没有收到 ACK 信息并且认为连接仍是打开的。一般来说，这个状态会持续 1 至 2 分钟。

例如请求端 A 号机发送一个初始序号（SEQ）987694419 给 1 号机。标志位 SYN 置为 1。服务器 B 号机收到这个序号后，将应答信号（ACK）和随机产生一个初始序号（SEQ）1773195208 发回到请求端 208 号机，因为有应答信号和初始序号，所以标志位 ACK 和 SYN 都置为 1。请求端 A 号机收到 B 号机的信号后，发回信息给 B 号机。标志位 ACK 置为 1，其它标志为都为 0。注意此时 SYN 值为 0，SYN 是标示发起连接的，上两步连接已经完成。

16 位窗口大小：TCP 的流量控制由连接的每一端通过声明的窗口大小来提供。窗口大小为字节数，起始于确认序号字段指明的值，这个值是接收端正期望接收的字节。窗口大小是一个 16 字节字段，因而窗口大小最大为 65535 字节。新的窗口刻度选项，允许这个值按比例变化以提供更大的窗口。TCP 的流量控制由连接的每一端通过声明的窗口大小来提供。

16 位检验和：检验和覆盖了整个的 TCP 报文段：TCP 首部和 TCP 数据。这是一个强制性的字段，一定是由发端计算和存储，并由收端进行验证。TCP 检验和的计算和 UDP 检验和的计算相似，使用伪首部帮助计算。

16 位紧急指针：只有当 URG 标志置 1 时紧急指针才有效。紧急指针是一个正的偏移量，和序号字段中的值相加表示紧急数据最后一个字节的序号。TCP 的紧急方式是发送端向另一端发送紧急数据的一种方式。优先指针指向后面是优先数据的字节。

选项：最常见的可选字段是最长报文大小，又称为 MSS (Maximum Segment Size)。每个连接方通常都在握手的第一步中指明这个选项。它指明本端所能接收的最大长度的报文段。选项长度不定,但长度必须以字节记。

在一个连接建立和一个连接终止时，双方交换的报文段仅有 TCP 首部。如果一方没有数据要发送，也使用没有任何数据的首部来确认收到的数据。在处理超时的许多情况中，也会发送不带任何数据的报文段。

TCP 是一个面向连接的协议。无论哪一方向另一方发送数据之前，都必须先在双方之间建立一条连接。建立连接的过程就是三次握手的过程。其步骤如下:

1)请求端 A 号机发送一个初始序号（SEQ）987694419 给 B 号机。

2)服务器 B 号机收到这个序号后，将此序号加 1 作为应答信号（ACK），同时随机产生一个初始序号（SEQ）1773195208，这两个信号同时发回到请求端 A 号机，意思为：“消息已收到，让我们的数据流以 1773195208 这个数开始。”

3)请求端 A 号机收到后将确认序号设置为服务器的初始序号（SEQ）1773195208 加 1 为 1773195209 作为应答信号。

以上三步完成了三次握手，双方建立了一条通道，接下来进行数据传输。

建立一个连接需要三次握手，而终止一个连接要经过 4 次握手。这是因为一个 TCP 连接是全双工（即数据在两个方向上能同时传递），每个方向必须单独地进行关闭。4 次握手实际上就是双方单独关闭的过程。其步骤如下：

1)A 号机将 FIN 置 1 连同序号(SEQ)987695574 发给 B 号机请求终止连接。

2)B 号机收到 FIN 关闭请求后，发回一个确认，并将应答信号设置为收到序号加 1，这样就终止了这个方向的传输。

3)B 号机将 FIN 置 1 连同序号(SEQ)1773196056 发给 A 号机请求终止连接。

4)A 号机收到 FIN 关闭请求后，发回一个确认，并将应答信号设置为收到序号加 1，至此 TCP 连接彻底关闭。

有关发起和终止 TCP 连接的规则都能从下图 4-6 所示的状态变迁图中得出:

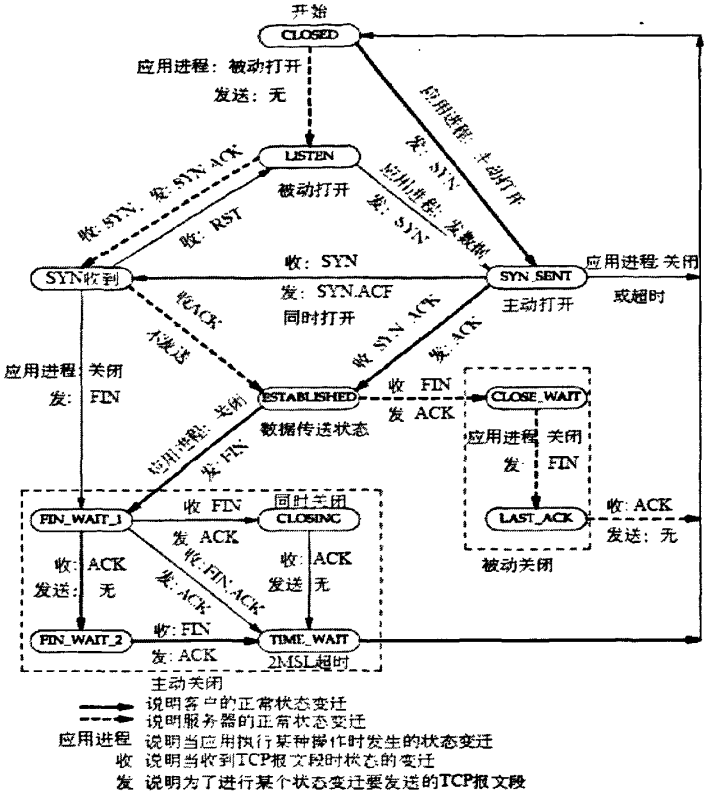

图 4-6 TCP 状态转移图

在这个图中要注意的第一点是一个状态变迁的子集是“典型的”。我们用粗的实线箭头表示正常的客户端状态变迁，用粗的虚线箭头表示正常的服务器状态变迁。许多流行的应用程序如 Telnet、Rlogin、FTP 和 SMTP 都使用 TCP。

TCP 协议对这些应用层协议规定了整数标志符，称为端口序号。被规定的端口序号成为保留端口，其值在 0～1023 范围内（如端口序号 23，用于远程终端服务）。此外还有自由端口序号，供个人程序使用，或者用来区分两台主机间相同应用层协议的多个通信，即两台主机间复用多个用户会话连接。

用户会话连接都有一个插口序号，由主机的 IP 地址和端口序号组成。插口序号是惟一的，一对插口序号惟一标识一个端口的连接。利用插口序号可在目的主机中区分不同源主机对同一个目的主机相同端口序号的多个用户会话连接。

在 TCP 协议段的头部各域中具有码位项。SYN 码位为应用数据流的开始位，FIN 码位为应用数据流的结束位。因此可利用 SYN/FIN 两个码位来规定某一应用报文（或某一应用数据流）的开始与结束。TCP 协议就是利用端口序号和 SYN/FIN 码位来区分应用数据流并判断其性质的，从而使具有四层功能的高端路由器具有某些对应用数据流的控制功能。

第五章 实例的验证与测试

5.1 基于单片机的数字电视机顶盒接入 Internet 技术

5.1.1 硬件平台介绍

网络部分硬件设计结构框图如图5-1所示：

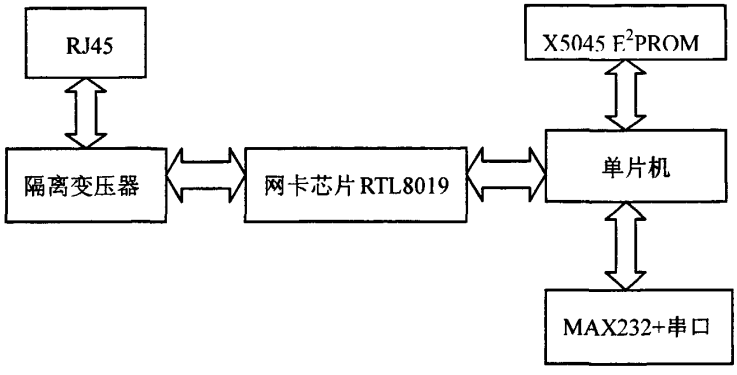

图5-1 硬件结构框图

在该系统中主要包括MCU单元，串口通讯单元，EEPROM单元和网卡芯片单元等。

在系统中还使用x5045作为外部扩展的EEPROM，用来存储IP地址、物理地址以及网卡的其他配置信息。同时x5045还具有电压监控，看门狗定时器，上电复位三种功能，使用x5045监控系统的运行过程，当系统不稳定的时候可以进行有效的复位。

由于单片机的高低电平和串行口的高低电平不一样，二者之间需要进行电平转换，在本设计中使用MAX232作为串口电平转换芯片，负责单片机与PC机的电平转换。数据可以从串口输入到单片机，单片机在把数据送到8019传出去。反之，从8019接收到的数据通过单片机可以从串口发出去。

网卡芯片采用的是Realtek公司生产的RTL8019AS以太网卡控制器，价格低廉，和NE2000兼容。RTL8019AS符合ETHERNET II和802．3标准；内置16K SRAM，用于收发缓冲，降低对主处理器的要求；支持8/16位数据总线，8个中断申请，16个I/O基地址选择等许多特性；在本设计中选用8位数据总线，选择8位

数据总线的方法是在RTL8019AS的IOCS16B引脚接一个27K的下拉电阻[12]。

网卡控制器与以太网是不能直接相连的，中间要通过网络隔离变压器才能连接到以太网上。网络隔离变压器的作用是隔直通交，避免双绞线上的直流电平干扰芯片（RTL8019AS）的工作点，利用变压器自身的通频带限制高频的干扰。

网卡芯片 RTL8019AS 是高度集成以太网控制器，它能够简单的解答即插即用 NE2000 兼容适配器，这种适配器具有二重和功率下降特性。通过三电平控制特性，RTL8019AS 是已制的对网络设备 GREEN PC 理想的选择。全二重功能能够模拟传播和接收在双绞线到全二重以太网交换机。这个特性不仅强带宽从 10 到 20MBPS，而且避免了由于以太网频道争夺特性导致的读出多路存取协议的问题。微软公司的即插即用功能能减轻用户较差的营业收入而注意适配器资源，如 IRQ,输入输出，和存储器地址等等。为了提供完全解决即插即用方案，RTL8019AS 在集成 10BASET 收发器，BNC,和 AUI 接口之间的自动检测功能。此外，8 条 IRQ 总线和 16 条基本地址总线为大资源情况下提供了宽松的环境。RTL8019AS 支持 16k，32k，和 64k 字节 BROM 和闪存接口。它仍然提供页面模式功能，这种功能能支持在仅 16k 字节内存系统空间下的 4M 字节的 BROM. 此外，BROM 的无用命令被用来释放 BROM 内存空间。RTL8019AS 用 16k 字节 SRAM 设计在单片芯片上，它的设计不仅提供了更多友好的功能，而且节省了 SRAM 存储资源[13]。

8019提供3种配置I/O端口和中断的模式：第一种为跳线模式（Jumper），RTL8019AS的I/O端口和中断由跳线引脚决定；第二种为即插即用模式（Plug and Play，PnP），由软件自动配置；第三种为免跳线模式（Jumperless），RTL8019AS的I/O端口和中断由9346（EEPROM）里的配置信息决定。在本电路中，使用x5045作为闪盘存储MAC地址和其他配置信息。PnP模式主要使用在PC机中。所以，使用跳线模式来选择I/O端口和中断。RTL8019AS第65引脚JP接高电平（直接接到VDD或通过一个10kΩ的电阻上拉），8019工作在跳线模式。I/O端口基址选为300H，中断使用IRQ2/9引脚。I/O端口的基地址由单片机和8019之间的接线决定，如果P2.6=8019CS，低电平有效，则IO_BASE_ADDRESS＝0xBF00。此外8019还具有8个IRQ接口，本系统中没用到IRQ，采用查询方式。

当系统上电复位后，在RSTDRV下降沿，8019AS读入各个跳线引脚的状态，写入到系统配置寄存器中，作为系统默认的初始配置。

8019 输入输出地址共 32 个，地址偏移量为 00H--1FH（对应于 BF00H－－BF1FH）。其中 00H－－0FH 共 16 个地址，为寄存器地址，寄存器分成 4 页 PAGE0--PAGE3，与 NE2000 兼容的寄存器只有 3 页(Page0-Page2)，为了保证驱动程序对所有 Ne2000 的网卡有效，不要去操作第四页的寄存器。10H－－17H

共 8 个地址，为 DMA 地址。18H－－1FH 共 8 个地址，为软复位端口。8019 的硬件复位很简单，只需在上电时对 RSTDRV 输出一高电平就可以了。8019 复位的过程将执行一些操作，比如将 93c46 读入，将内部寄存器初始化等，至少需要 2 毫秒的时间。推荐等待更久的时间之后才对网卡操作，比如 100 毫秒之后才对它操作，以确保完全复位。

ICS16B=LOW 时采用 8 位 DMA 操作模式，上面的地址中只有 18 个是有用的： 00H－－0FH 共 16 个寄存器地址。10H DMA 地址 （10H－－17H 的 8 个地址是一样的，都可以用来做 DMA 端口，只要用其中的一个就可以了）。1FH 复位地址（18H 到 1FH 共 8 个地址都是复位地址，每个地址的功能都是一样的，只要其中的一个就可以了，但实际上只有 18H、1AH、1CH、1EH 这几个复位端口是有效的，其他不要使用，有些兼容卡不支持 19H、1BH、1DH 等奇数地址的复位）。

从程序员的角度来说，对 8019 的操作是比较简单的，驱动程序只需要将要发送的数据按一定的格式写入芯片并启动发送命令，8019 会自动把数据包转换成物理帧格式在物理信道上传输。反之，8019 收到物理信号后将其还原成数据，按指定格式存放在芯片 RAM 中以便主机程序取用。简言之就是 8019 完成数据包和电信号之间的相互转换：数据包<===>电信号。以太网协议由芯片硬件自动完成，对程序员透明。驱动程序有 3 种功能：芯片初始化、收包、发包。

RTL8019AS 内部有两块 RAM 区。一块 16K 字节，地址为 0x4000～0x7fff；一块 32 字节，为寄存器区，地址为 0x0000～0x001f。RAM 按页存储，每 256 字节为一页。一般将 RAM 的前 12 页（即 0x4000～0x4bff）存储区作为发送缓冲区；后 52 页（即 0x4c00～0x7fff）存储区作为接收缓冲区。寄存器分为 4 页：PAGE0、PAGE1、PAGE2、PAGE3，由 RTL8019AS 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 RTL8019AS 自己定义的，对于其他兼容 NE2000 的芯片如 DM9008 无效。远程 DMA 地址包括 10H～17H，都可以用来做远程 DMA 端口，只要用其中的一个就可以了。复位端口包括 18H～1FH 共 8 个地址，功能一样，用于 RTL8019AS 复位[14]。

接收和发送数据包都必须通过 DMA 读写网卡内部的 16K RAM，网卡的 16K RAM 是一个双端口 RAM，所谓双端口就是有两套总线连接到该 RAM 上，一套总线是单片机读写网卡上的 RAM.即远程 DMA；另一套总线是网卡控制器读写网上的 RAM，即本地 DMA[15]。

1 发送子程序

数据包的发送过程包括三个步骤：主处理器将数据包按照 RTL8019AS 发送

数据帧格式进行封装;封装完之后通过远程 DMA 通道将数据包送到 RTL8019AS 的发送缓冲区；然后通过本地 DMA 将数据送到 FIFO，通过设置寄存器 CR 启动发送。RTL8019AS 完成上一帧的发送，在开始下一帧的发送。

2 接收子程序

接收数据是通过本地 DMA 从网卡接口接收.RTL8019AS 对接收到的数据包通过 MAC 比较，CRC 校验，由 FIFO 存到接受缓冲区，收满一帧后以中断或者寄存器标志方式通知主处理器，主处理器通过远程 DMA 将缓冲区的数据读到自己的内存中进行处理[16]。

5.1.2 软件平台介绍

随着单片机开发技术的不断发展，从普遍使用汇编语言到逐渐使用高级语言开发，单片机的开发软件也在不断发展，Keil 开发软件是目前最流行的开发 MCS-51 系列单片机的软件,Keil 提供了包括C 编译器、宏汇编、连接器、库管理和一个功能强大的仿真调试器等在内的完整开发方案，通过一个集成开发环境（μ'Vision）将这些部份组合在一起[17]。

μ Vision2 是Keil Software 的一个新的IDE，它结合了项目管理、生成工具、源代码编辑、程序调试和在一个强大的环境中完全模拟。μ Vision2 提供了一个简单易用的开发平台帮助您使程序运行得比以前更快。编辑器和调试器集成到一个应用程序中，并提供一个无缝的嵌入式项目开发环境。

μ Vision2 提供了下面这些独特的功能：

- 器件数据库：自动为您选择的芯片设置汇编器、编译器和连接器选项。这就使您节省了配置工具的时间，并帮助您更快地编写代码；
- 健壮的项目管理器：可以在一个项目文件中对目标创建几个不同的配置。只有Keil μ Vision2 IDE 允许创建一个用于模拟的输出文件、一个用于仿真器调试的输出文件和一个向EPROM 编程的输出文件，上面的这些文件都来自于同一个项目文件[18]；
- 自动独立生成的集成生成工具。不用指出哪个头文件和包含文件由哪个源文件使用，Keil 编译器和汇编器会自动完成这项工作；
- 交互的错误纠正。在编译项目时，错误和警告信息在输出窗口显示。当μ Vision2仍然在后台编译时，你可以纠正项目文件中的错误。错误和警告相关的行号在对源文件作了修改后会自动重新同步。

安装 CQKJ 仿真器下载插件 V3.8, 就可以下载程序了。

PC 端程序运行环境选用的是 Visual Basic 6.0，Visual Basic 是由微软公司推

出的一套完整的 Windows 系统软件开发工具，可用于开发 Windows 环境下的各类应用程序，是一种可视化、真正面向对象、采用事件驱动方式的结构化高级程序设计语言和工具的完美集成。它编程简单、方便、功能强大，具有与其它语言及环境的良好接口，不需要编程开发人员具备 C/C++或者 Turbo Pascal 语言知识和特别高深的专业知识，只要懂得 Windows 的界面及其基本操作，就可以迅速上手，而 VB 在程序界面设计、多媒体开发方面更是独具优势[19]。

现在比较流行的版本是 Visual Basic 6.0,它继承了旧版本 Visual Basic 的所有优点，同时增强了功能和简化了操作界面。Visual Basic 不仅仅是一种语言，而且是一个集应用程序开发，测试和调试等功能为一体的集成式开发环境。下面将对这个开发环境进行一些介绍。首先确认系统已经安装 VB6.0 企业版，移动鼠标到屏幕左下角，顺序点击 “开始->程序->Microsoft Visual Studio 6.0->Microsoft Visual Basic 6.0”，这样就可以启动 VB 了.启动后提示你选择要建立的项目类型(VB 中一个应用程序称为一个项目)[20]，如图 5-2 所示：

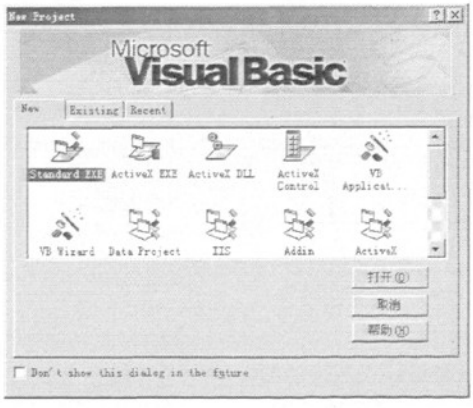

图 5-2 启动 VB 界面

图 5-2 所示窗口左上方有三个选项卡，每个选项卡的作用分别是：

New-----创建一个新的工程。

Existing-----打开已经保存在机器上的工程。

Recent-----打开最近编辑过的工程。VB 按时间顺序自动记录你最近过打开的工程。

选择 Standard EXE(标准 EXE 程序),然后点击“打开(O)”按钮，就可进入 VB 开发界面了[21]。

5.1.3 具体实现

下面先介绍一下单片机端的程序:

1、初始化

初始化部分包括 8019AS 的初始化、STT89E516RD 定时器初始化、系统内存空间分配等。

8019AS 的初始化如下：

- 复位 8019，再使网卡处于停机模式，延时 10ms 确保芯片处于停机模式；
- 设置芯片的发送缓冲区和接收缓冲区大小。0x40-0x4B 为网卡的发送缓冲区，共 12 页，刚好可以存储 2 个最大的以太网包；使用 0x4c－0x7f 为网卡的接收缓冲区，共 52 页；
- 设置芯片 MAC 地址；
- 设置芯片工作于 8 位 DMA 工作方式；
- 设置其他工作寄存器，然后使芯片处于正常工作模式。

2、以太网收发

对以太网包的收发是实现该系统功能的基础，更高层数据通信协议 ARP、IP、ICMP、TCP、UDP 的实现都必须以这个为基础。在本系统中，以太网的数据包格式采用 802。3。它的帧结构如图 5-3 所示。

前导位PR	帧起始位置SD	目标MAC地址	源MAC地址	类型TYPE	数据DATA	填充PAD	校验FCS

图 5-3　802.3 帧结构

物理信道上的收发操作均使用这个帧格式。其中，前导序列、帧起始位、CRC 校验由硬件自动添加/删除，与上层软件无关。收到的数据包格式并不是 802.3 帧的真子集，而是如图 5-4 所示。

接受状态	下一页指针	以太网帧长度	目标MAC地址	源MAC地址	类型TYPE	数据DATA	填充PAD	校验FCS

图 5-4　8019AS 接收帧结构

8019 自动添加了“接收状态、下一页指针、以太网帧长度(以字节为单位)”三个数据成员。发送数据包的格式是 802.3 帧的真子集，如图 5-5 所示：

目的MAC地址	源MAC地址	类型TYPE	数据DATA	填充PAD

图 5-5 8019AS 发送帧结构

将待发送的数据按发送的帧格式封装，通过远程 DMA 通道送到 RTL8019AS 中的发送缓存区，然后发出传送命令，就完成帧的发送。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，数据写入 RTL8019AS 的 RAM，再启动本地 DMA，将数据发送网上。

在本系统中，通过对 CURR 是否为 BNRY+1 来判断是否收到新的数据包。CURR 是网卡写内存的指针，它指向当前正在写的页的下一页，网卡写完接收缓冲区一页，就将这个页地址加一。BNRY 要由用户来操作。用户从网卡读走一页数据，要将 BNRY 加一，然后再写到 BNRY 寄存器。初始化是将 CURR=BNRY+1。当收到新的包时，上述条件不成立，因此查询上述两个寄存器可以判断是否收到新的数据包。当收到新的数据包时，先通过 DMA 方式读出包的前 18 个字节，该 18 个字节包括读状态、包长度、下页地址、发送/接收端 MAC 地址，以及协议类型。通过这些信息可以决定该包是否有效、将包交给哪个高层协议处理等。

3、ARP 及 PING 的实现

ARP 即“地址解析协议”，本质是完成 32 位 IP 地址到以太网 48 位物理地址的映射。考虑到 51 单片机资源有限，ARP 协议实现了学习、老化、更新、溢出等算法。PING 程序所用到的协议为 ICMP 协议。完整 ping 过程主要用 4 个函数实现。Ping 请求(PingRequest（）)、Ping 应答(PingAnswer（）)、收到应答后回显(PingEcho（）)、定时操作(PingCycle（）)，各个函数的功能如图 5-6 所示：

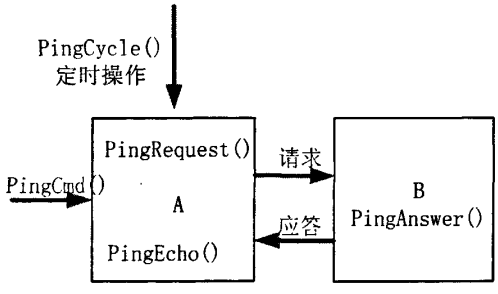

图 5-6 A ping B 过程

PingRequest 完成 Ping 请求。当输入 Ping 命令后，调用此函数，发出请求包。PingAnswer 完成 Ping 应答。每当收到发给自己 IP 的请求包就自动应答。

PingEcho 显示应答信息。每当收到应答就立即显示相关信息。

PingCycle 定时操作 pingbuf 记录缓冲区。每 100 毫秒钟扫描一次，并根据当前情况和状态进行状态变迁[22]。

在发出 Ping 命令后，为了不使处理器一直在等待回应而浪费 CPU 资源，本系统中设计了一个 pingbuf 记录缓冲区，如图 5-7 所示。

状态	次数	发送区指针

图 5-7 pingbuf 记录缓冲区格式

此表 100 毫秒钟被查询一次，根据当前情况改变状态。它包括状态、次数、发送区指针三个域。“次数”字段指示测试剩余数，为 0 表示测完，同时改变状态为 0 以表明此记录项现在空闲。“发送区指针”字段保存 ICMP 报文缓冲区首址指针。“状态”字段记录状态号，含义如下[23]：0---空闲；1---已发出但无应答；2---已发出且应答；3---等 ARP；4---第一次准备发(用于同步 100 毫秒时钟)。

状态变迁图如图 5-8 所示，其中每个箭头表示 PingCycle()扫描一次[24]。

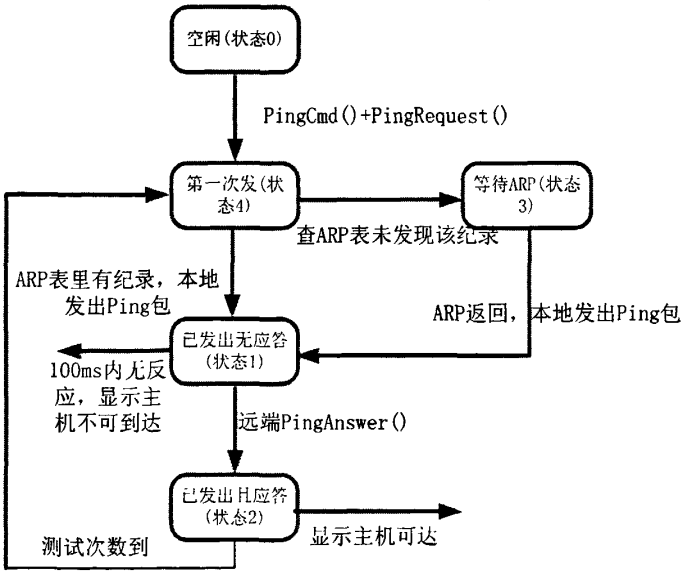

图 5-8 pingbuf 的状态变迁图

4、 以太网部分主程序框图

以太网处理部分主程序 maincycle（）框图如图 5-9 所示。

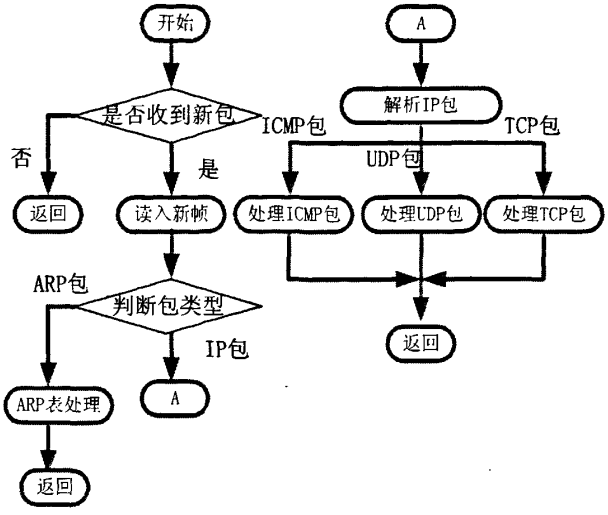

图 5-9 maincycle（）框图

主程序框图如图 5-10 所示。

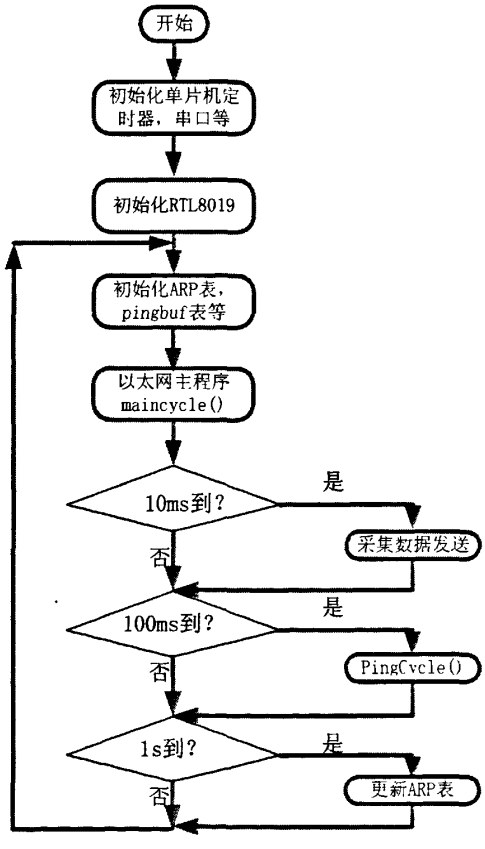

图 5-10 主程序框图

5、API 说明

共提供了四个 API 函数，后续函数将提供 TCPSocket()，TCPConnect（），User_TCPSend（），TCPClose（）。这四个函数功能如下：

1．TCPSocket()，申请一个套接字。套接字总的个数在 tcp.h 里的 NO_CONNECTION 定义，在默认代码中，只定义了一个。该函数返回值为套接字序号，如果返回为 ERR_SOCK_FULL，说明所有套接字已被使用。因此，在编写应用程序时，应判断是否申请到套接字。

2．TCPConnect（），向指定 IP、端口连接。原型如下：

void TCPConnect(unsigned char sock, union IP_address destip, unsigned int destport, void (* recv)(unsigned char xdata *buf ,unsigned int size),void (* close)(),void (* connect)())

unsigned char sock：套接字编号（用于识别哪个套接字）；

union IP_address destip：对方的 IP 地址；

unsigned int destport：对方端口号；

void (* recv)(unsigned char xdata *buf ,unsigned int size)：套接字收到数据后调用的回调函数指针，该指针指向的函数需自己定义。在 tcp 建立连接后，如果这个套接字收到数据，则会调用这个函数。buf 为数据首地址，size 为数据长度。在编写应用程序时，比如应用程序要把收到的数据发往串口，则在这里把 buf 指向的数据发往串口。注意：这个函数必须为 reentrant 类型。

void (* close)()：该套接字关闭后调用的回调函数指针，同上，该指针指向的函数必须自己定义。这个函数用于的应用层，当连接关闭时调用这个函数，比如在这里把应用层的连接标志位清掉，表示连接已关闭。注意：这个函数必须为 reentrant 类型。

void (* connect)():到套接字发送连接请求后，连接上时会调用该函数指针指向的函数。该指针指向的函数必须自己定义。注意：这个函数必须为 reentrant 类型。

3. User_TCPSend（）原型为：

unsigned char User_TCPSend(unsigned char sock,unsigned char xdata *buf,unsigned int size)

该函数用于用户发送数据。

unsigned char sock：套接字编号；

unsigned char xdata *buf：要发送数据的缓冲区首地址；

unsigned int size：数据长度。

如果发送成功，则返回 FALSE，如果发送失败，则返回相应错误类型。

ERR_BUF_FULL 表示套接字缓冲区满，ERR_NO_ESTABLISH 表示套接字未建立连接。在调用此函数时需要进行返回值判断[25]。

4. TCPClose（）该函数原型为：

void TCPClose(unsigned char sock)

该函数用于用户主动关闭一个套接字。

unsigned char sock：套接字编号（用于识别哪个套接字）。

下面介绍一下,发送端的软件实现，用 Visual Basic 6.0 编写的发送端的串口设置软件如图 5-11 所示：

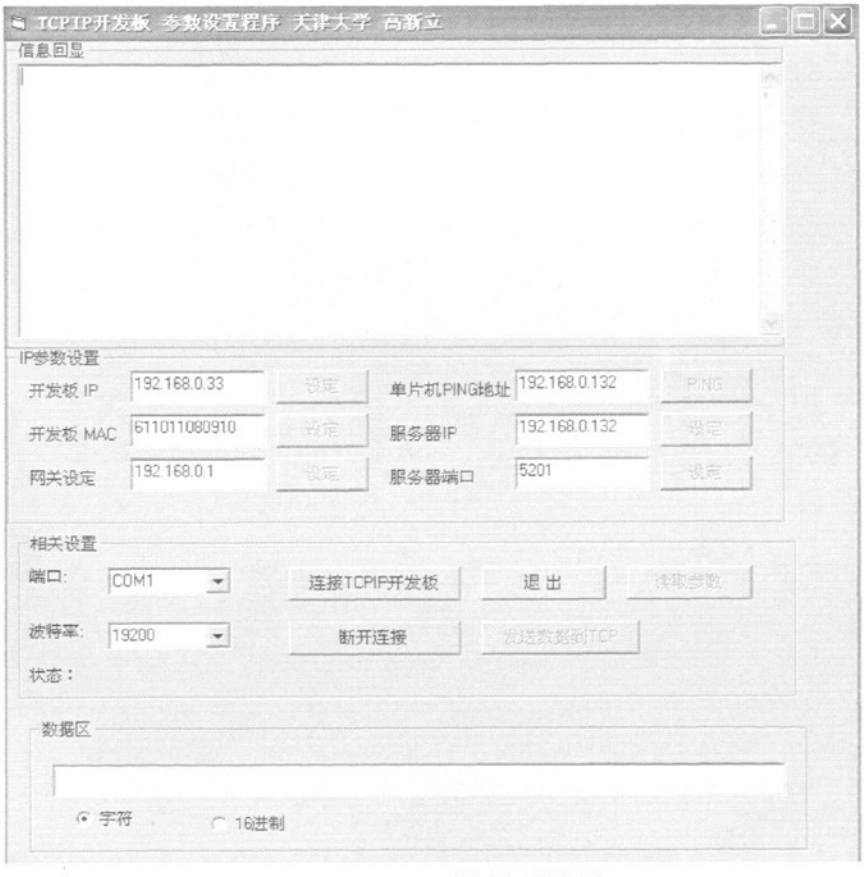

图 5-11 TCPIP 开发板参数设置

开发板的参数设置为如下所示：

IP 地址：192.168.0.33　　网关地址：192.168.0.1

服务器IP：192.168.0.132　　端口号：5201

同时设置好选择哪个串口以及串口的波特率是多少,我选用的是串口1,波特率设置为19200.然后把PC 的IP 设为：192.168.0.132，在运行中输入：cmd，再输入：ping 192.168.0.33–t就会显示如下图5-12所示界面：

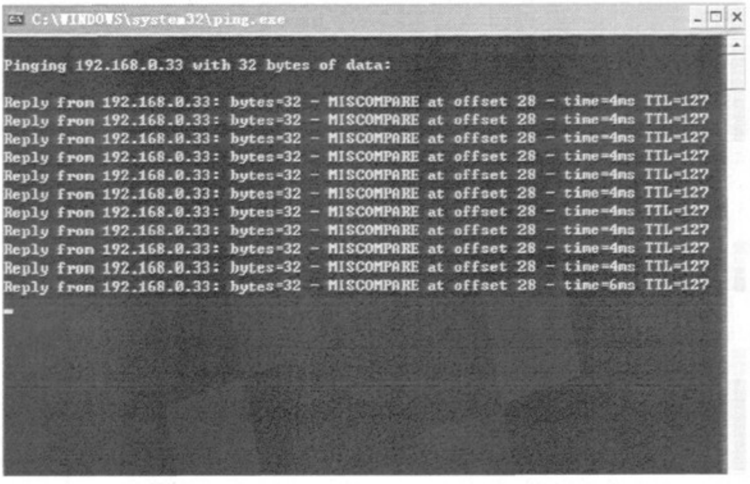

图5-12 Ping 界面

这是TCPIP.exe的运行界面，如图5-13所示：

图5-13 TCPIP.exe的运行界面

这里的端口号与设置的值一致，设置为5201，然后点击Start_Server，成功连接后有“连接成功”的提示，然后在串口设置的软件的数据区里面输入“TJU”，然后点击“发送数据到TCPIP”，在TCPIP.exe就会显示：“REV：TJU”。这样就完成了串口转TCPIP的功能了。

5.2 LWIP 协议在 OS20 操作系统中的移植

5.2.1 硬件设计介绍

STx5105平台是ST公司推出的一款数字电视机顶盒软硬件平台，它主要由主芯片STx5105、高频头、flash、DDR SDRAM组成，预留smart card接口，还包括了相应的软件，为广大机顶盒制造商提供了一个集成度高，低成本的解决方案。STx5105 是STx5105 接收方案的核心部件，用于标准清晰度数字电视的MPEG解码。此芯片采用嵌入设计，将32 位微处理器、TS 流解复用器、MPEG-2音频、视频解码器、中央DMA 控制器、诊断控制器、外部存储接口等集成在一起[26]。

芯片STx5105的结构框图，如下图5-14所示：

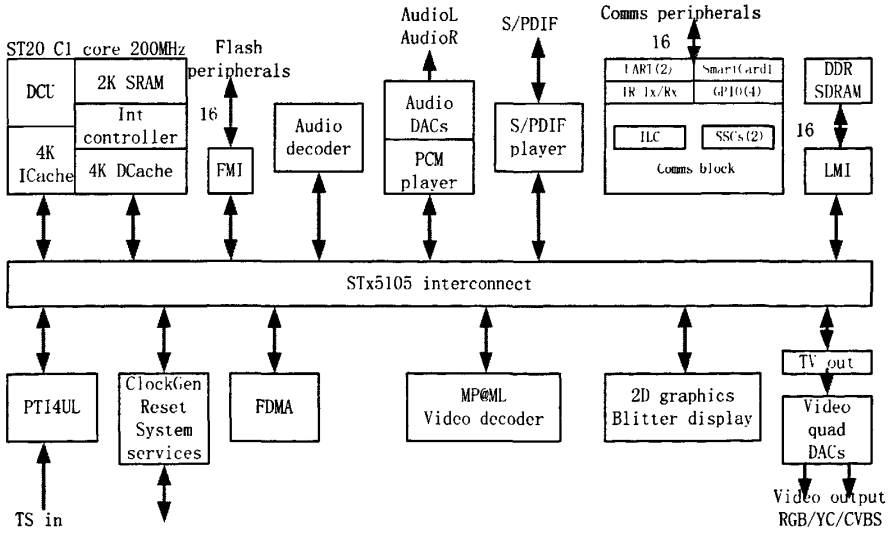

图5-14 STx5105结构框图

网卡芯片采用的是DM9000A，DM9000A实现以太网媒体介质访问层(MAC)和物理层(PHY)的功能，包括MAC数据帧的组装／拆分与收发、地址识别、CRC编码／校验、MLT-3编码器、接收噪声抑制、输出脉冲成形、超时重传、链路完

整性测试、信号极性检测与纠正等。

DM9000A的主要特性如下：

- 支持8 / 16位数据总线；
- 适用于10Base-T和100Base-T；
- 10 / 100 M自适应，适应不同的网络速率要求；
- 内置16 KB的SRAM，用于收发缓冲，降低对主处理器的速度要求；
- 与IEEE 802.3u兼容，支持IEEE802.3x全双工，可同时收发；
- 具有睡眠模式，可降低功耗；
- 采用48引脚LQFP封装，缩小PCB面积。

DM9000A的内部框图如图5-15所示：

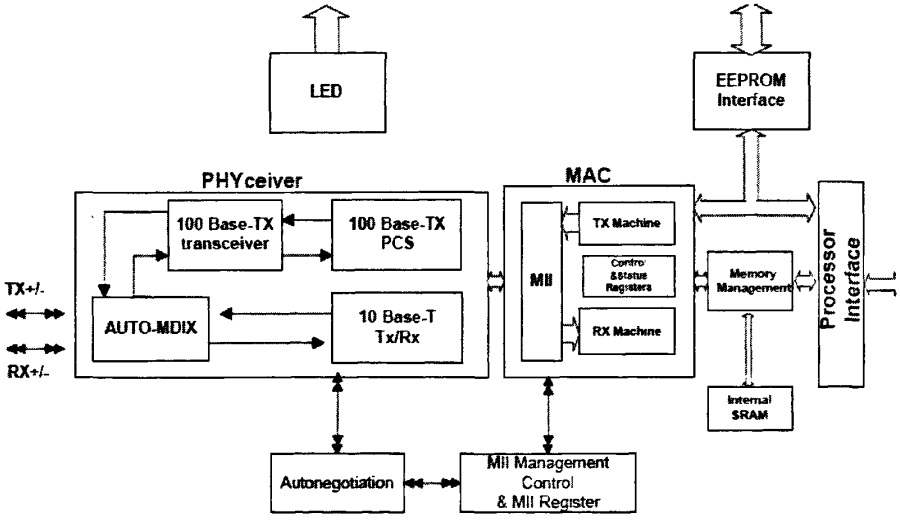

图5-15 DM9000A的内部框图

网络部分的连接框图如图5-16所示：

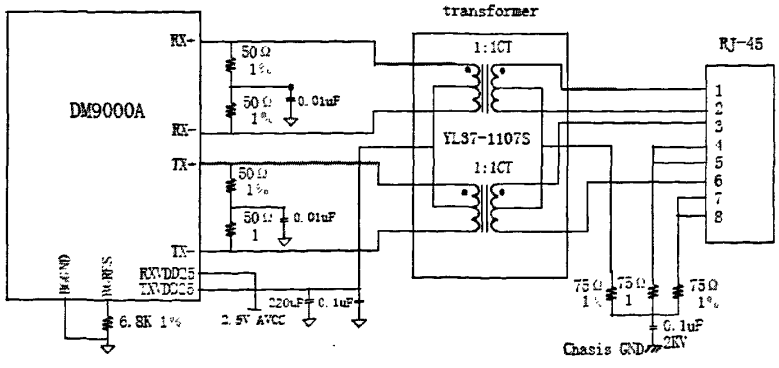

图5-16 网络部分的连接框图

系统总体的方框图如图5-17所示：

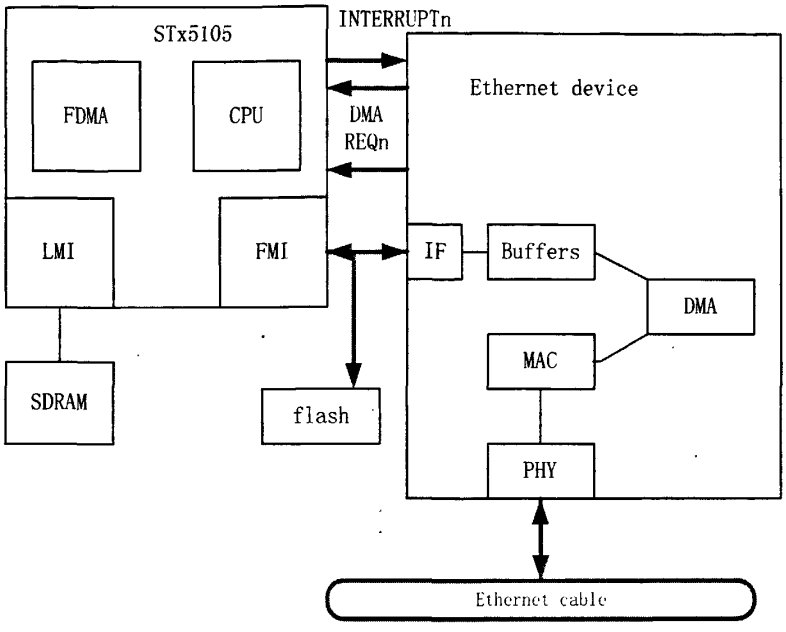

图5-17 系统整体框图

DM9000A可以和微处理器以8位或16位的总线方式连接，并可根据需要以单工或全双工等模式运行。在系统上电时，处理器通过总线配置DM9000A内的网络控制寄存器(NCR)、中断寄存器(ISR)等，以完成DM9000A的初始化。随后，DM9000A进人数据收发等待状态。

当处理器要向以太网发送数据帧时，先将数据打包成UDP或IP数据包，并通过8位或16位总线逐字节发送到DM9000A的数据发送缓存中，然后将数据长度等信息填充到DM9000A的相应寄存器内，随后发送使能命令，DM9000A将缓存的数据和数据帧信息进行MAC组帧，并发送出去。

当DM9000A接收到外部网络送来的以太网数据时，首先检测数据帧的合法性，如果帧头标志有误或存在CRC校验错误，则将该帧数据丢弃，否则将数据帧缓存到内部RAM，并通过中断标志位通知处理器，处理器收到中断后将DM9000A接收RAM的数据读出进行处理。

DM9000A自动检测网络连接情况，根据网速设定内部的数据收发速率是10Mbps或100Mbps。

DM9000A正常工作需要在上电后对内部寄存器进行初始化。该过程是通过FPGA对DM9000A外部控制总线和数据总线的读写操作完成的。具体流程如下：

1) 激活PHY

设置GPR(REG_1F) CEPI00 bit[0]=0；

由于复位后，DM9000A恢复默认的休眠状态以降低功耗，因此需要首先唤醒PHY。

2) 进行两次软复位，步骤如下：

设置NCR(REG_00)bit[2：0]=011；至少保持20μs；

清除NCR(REG_00)bit[2：0]=000；

设置NCR(REQ_00)bit[2：0]=011；至少保持20μs；

清除NCR(REG_00)bit[2：0]=000；

3) 配置NCR寄存器

设置NCR(REG_00)bit[2：1]=00；配置为正常模式。

通过改变该寄存器可以选择设置内部或者外部PHY、全双工或者半双工模式、使能唤醒事件等网络操作。

4) 清除发送状态

设置NSR(REG_01)bit[5]=1 bit[3]=l bit[2]=l；

5) 设置IMR寄存器(REG_FF)PAR bit[7]=l；启用RX / TX FIFO SRAM读 / 写地址指针自动返回功能。

6) 通过IMR寄存器(REG_FF)PRM bit[0] / PTM bit[1]，对RX / TX中断使能。如果需要在一个数据帧发送完后产生一个中断，应将PTM bit[1]置为1，如果需要在接收到一帧新数据时产生一个中断，应将PRM bit[1]置为1；

7) 设置RCR寄存器，使能数据接收功能。

以上步骤完成后，DM9000A已成功完成初始化。

首先，向DM9000A的发送缓存区中写入发送数据帧，写数据帧时需要先写入6字节的目的MAC地址，再写入6字节的源MAC地址，最后写入发送数据。随后，写入寄存器FCH和FDH，数据长度为16位，将高8位写入寄存器FCH，低8位写入寄存器FDH。最后，向DM9000A发出发送数据指令。DM9000A自动进行一些处理才将数据发送至以太网，包括：插入报头和帧起始分隔符；插入来自上层协议的数据，如果数据量小于64字节，则自动补齐64字节；根据目标地址、源地址、长度 / 类型和数据产生CRC校验序列，并插入校验序列位。处理完毕后，DM9000A即开始发送帧I。在帧I发送的同时，帧II的数据即可写入发送缓存区。在帧I发送完后，将帧II的数据长度写入寄存器FCH和FDH，最后将发送控制寄存器NSR(REG_01)的bit[1]置为高电平，即可开始帧II的发送。依此类推，下面发送的帧将会继续编号为帧I，帧II，帧I，帧II.....按照同样的方式发送。

DM9000A中的接收缓存区是一个环形结构，初始化后的起始地址为0C00H，

每帧数据都有4字节长的首部，然后是有效数据和CRC校验序列。首部4字节依次是01H、状态、长度低字节和长度高字节，如图5-18所示：

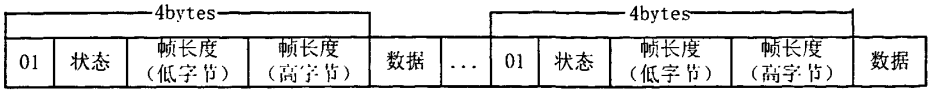

图5-18 帧结构

首部4字节含义如下：

第一个字节用于检测接收缓存区中是否有数据。如果这个字节是01 H，表明接收到了数据；如果为00H,则说明没有数据。但是如果第一个字节既不是01H，也不是00H，DM9000A就必须作一次软复位来从这种异常状态中恢复。

第二个字节存储以太网帧状态，由此可判断所接收帧是否正确。

第三和第四字节存储以太网帧长度。后续的字节就是有效数据。

接收过程如下：

查看中断状态寄存器。如果接收到新数据，寄存器ISR的PRS位将被置为0；

如果检测到PRS=0，清除PRS，STx5105开始读取接收缓存区数据。如果第一个字节是01H，则说明有数据；如果是00H，则说明无数据，需要进行复位；

根据获取的长度信息，判断是否读完一帧。如果读完，接着读下一帧，直到遇到首字节是00H的帧，说明接收数据已读完。STx5105可以重新查看中断状态寄存器，等待新的有效数据帧。

5.2.2 LWIP 协议介绍

LWIP是TCP/IP协议栈的一种实现。LWIP的主要目的是减少存储器利用量和代码尺寸，使LWIP适合应用于小的、资源有限的处理器如嵌入式系统。为了减少处理器和存储器要求，LWIP可以通过不需任何数据拷贝的API进行裁减。

正如其他TCP/IP协议的实现，分层协议的设计为LWIP的设计与实现提供一向导。每一个协议都作为一个模块来实现，提供一些与其他协议的接口函数。尽管各层分开实现，但是为了同时提高处理速度和内存利用两方面的性能，一些层在设计时违背这一原则。例如：当检验一接收到的TCP段（segment）的校验和（checksum）和分解TCP段时，源和目的IP地址必须被告知TCP模块。LWIP实现时不是通过函数调用把IP地址传递给TCP，而是TCP模块通过获取IP报头的结构进而自己提取这一信息[27]。

LWIP有几个模块组成，除了实现TCP/IP协议的各个模块(IP、ICMP、UDP、

和 TCP)，同时设计了许多支持模块。这些支持模块组成了操作系统模拟层、缓冲和存储管理子系统、网络接口函数和一些处理因特网校验和的函数。LWIP还包括关于API的摘要。

协议实现的过程模型以把系统划分成为不同的过程的方法进行描述。用于实现通讯协议的过程模型使每个协议作为孤立的过程运行。这种模型使用严格的协议分层，协议之间的通讯结点必须被严格定义。虽然这种方法有其诸多优势如协议能在运行时被增加，代码一般容易理解和调试，但也有不利因素。严格的分层，正如先前所述，并不总是实现协议的最好方法。同时，更重要的，每跨越一层，必须做一次上下文切换。这将意味着，接受一个TCP段要进行三次上下文切换：从网络接口的驱动，到IP处理，再到TCP处理，最终到应用处理。根据网络接口的设备驱动程序，对于IP过程，对于TCP过程和最后。在大多数操作系统中一个上下文切换所花的代价都是相当昂贵的。

另一个较普通的方法是把通信协议封装在操作系统的内核。在这种内核实现通讯协议的情况下，应用程序通过系统调用完成通讯。通讯协议之间不严格区分，但可以使用交叉协议分层技术。

LWIP所使用的过程模型是：把所以协议封装到一个过程中，与操作系统内核分开。应用程序可能也驻留在LWIP处理过程中，或者在单独的过程中。TCP/IP栈和应用程序之间的通信可以通过函数调用实现，也可以通过更为抽象的API。

以上两种LWIP的实现方法各有其优缺点。把LWIP作为一个过程的主要优点是便于在不同的操作系统上移植。由于LWIP的设计目标是面向小的操作系统，这些操作系统一般不支持进程外交换（swapping out processes）或者虚拟存储，这样由于LWIP处理过程交换或者翻页到磁盘而引起的不得不等待磁盘响应造成的延迟将不再是一个问题。尽管在获得服务响应前必须等待调度仍然是一个问题，但是，在LWIP设计时，这并没有妨碍它在一操作系统内核中实现[28]。

通讯系统中的存储和缓冲管理必须能够适应大小变化的缓冲区，从几百字节的包含完全大小TCP段的缓冲区到仅仅包含几个字节的短的ICMP回报。而且，为了避免拷贝它应当尽可能让缓冲区的数据内容驻留在内存中，网络子系统不管理像应用存储或ROM这样的内存。

Pbuf在LWIP的内部表示一包，也是为了最小限度的使用栈这一特殊需要而设计。Pbufs类似于用于BSD实现的mbufs。pbuf结构既支持分配动态内存来保存包内容，也支持把包数据存储在静态存储区。Pbufs能在一张列表中一起被连在一起，称为一个pbuf链，这样一个包可以跨越若干个pbufs。

Pbufs具有三种类型，PBUF RAM，PBUF ROM，和PBUF POOL。图5-19中pbuf描绘了PBUF RAM类型，和储存的被pbuf子系统管理的数据包。图5-20中的

pbuf是被链在一起的pbuf的一个例子，在其中链的第一个pbuf具有PBUF RAM类型(where the first pbufin the chain is of the PBUF RAM type)，而第二个具有PBUF ROM类型，这意味着它具有不被pbuf系统管理的存储数据。第三种类型的pbuf ，PBUF POOL如图5-21所示，包括从共有的固定大小的pbufs分配的固定大小的pbufs（consists of fixed size pbufs allocated from a pool of fixedsize pbufs.）。一个pbuf链可能包括多重类型的pbufs[29]。

PBUF_POOL主要被网络设备驱动程序使用，因为对操作系统来说分配单一的pbuf速度较快并且适合用于中断管理。当应用程序发送位于被应用程序管理的存储区的数据时，PBUF_ROM被使用。在pbuf被移交到TCP/IP栈后，数据不能修改，因此这一pbuf类型，这类型主要用于数据位于ROM时(因此名称为PBUF_ROM)。PBUF_ROM pbuf中的数据可能会用到的头存储在PBUF_RAM pbuf中，它链接在PBUF_ROM pbuf的前面，如图5-20所示。

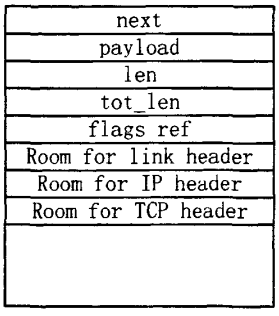

图5-19 PBUF RAM类型

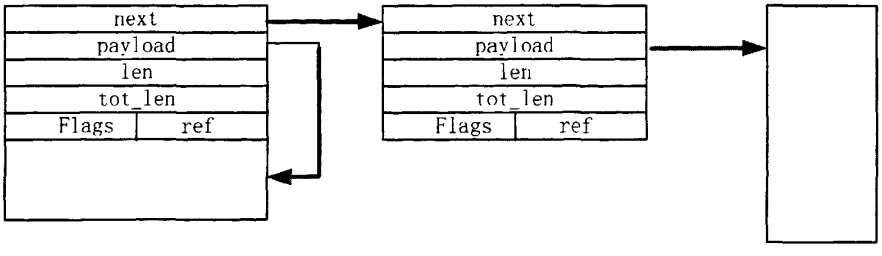

图5-20 pbuf被链在一起

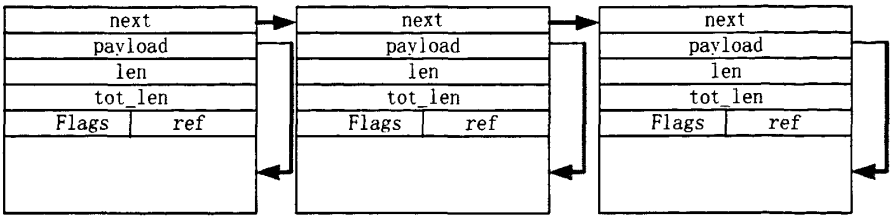

图5-21 PBUF_POOL 链

当应用程序发送动态地产生的数据时，PBUF_RAM类型的Pbufs也被使用。在这种情况下，pbuf系统不光为应用数据分配存储空间，也为将要发送的数据的报头准备空间。如图5-19所示。pbuf系统不能预先知道为将要发送的数据准备什么样的报头，并且假定最坏的情况。报头大小在编译时可动态配置。

实质上,进入pbufs的是PBUF_POOL类型，离开pbufs的是PBUF_ROM或PBUF_RAM类型。pbuf结构包括两个指针，两长度域，一个flags域，和一参考计数。pbuf链中next域是一个指向下一个pbuf的指针。Payload指针指向pbuf中的数据的起始位置。len域包含pbuf的数据内容的长度。Tot_len域包含当前的pbuf的长度和在pbuf链中接下来的pbufs的所有len领域的总数。换句话说，tot_len域是len域和pbuf链中的随后的pbuf中的tot_len域的值的总和。flags域表明pbuf的类型，而ref领域包含一参考计数。 Next和payload域是内部指针和依赖于处理器体系结构的数据大小。两个长度域为16位无符号整形，flags和ref域均为4bit宽。pbuf结构整个的大小取决于所使用的处理器体系结构中一个指针的大小及可能的最小alignment的大小。在带有32位指针和4个字节alignment的体系结构，整个的大小为16字节，16位指针和1个字节alignment的体系结构上，大小是9个字节。

pbuf模块为操纵pbufs提供了函数。函数pbuf_alloc()完成分配一个pbuf的任务，它能够分配上面所说的三种pbuf中的任何一种。pbuf_ref()增加参考计数。pbuf_free()完成释放分配的工作，它首先减少pbuf的参考计数。如果参考计数到达零表示pbuf已经被释放。函数pbuf_realloc()收缩pbuf使它刚好能够包含数据大小。pbuf_header()调整payload指针和长度域，以便对pbuf中的数据的报头进行预先估计。buf_chain()和pbuf_dechain()用于用链接pbufs。

支持pbuf调度的存储管理非常简单。它处理内存中连续区域的分配和释放，可以紧缩一个预先分配的内存块。内存管理器使用系统中总内存的专用部分，这确保网络系统不会使用所有可利用内存，而且如果网络系统用了所有它自己的内存其他程序的操作也不会影响它。在内部，内存管理通过将一种小的结构放置在每一被分配的内存块的顶端上来追踪分配的内存。这个结构中设置两个指针指向内存中下一个和前一个分配块，还有一个used标志用来指示这个分配块是否已经

被分配。

通过搜索一个未使用的内存块来分配内存，这个内存块对于请求分配来说足够大。使用最先适用原则，因此第一块被使用的内存足够大。当一个分配块释放时，used标志被设为0。为了防止碎片，检测下一个和上一个分配块的used标志，如果它们还没被使用，几个块合并成一个大未使用块，如图5-22所示：

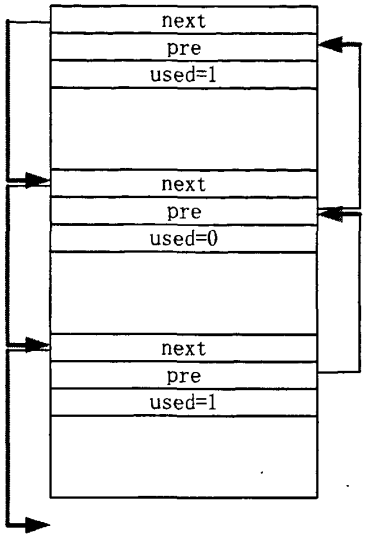

图5-22 内存分配结构

网卡结构体的结构如下所示：

```
struct netif{
        struct netif *next;
        char name[2];
        int num;
        struct ip_addr ip_addr;
        struct ip_addr netmask;
        struct ip_addr gw;
        void (*input)(struct pbuf *p, struct netif *inp);
        int (* output)(struct netif *netif, struct pbuf *p,struct ip_addr *ipaddr);
        void *state;
            }
```

当发送和接收包时，三个IP地址ip_addr，netmask和gw由IP层使用。给网络接口配置多于一个IP地址是不允许的，一个网络接口应当为每一个IP地址创建。当包接收到时，设备驱动应当调用input指针指向的函数。

网络接口通过output指针连接到设备驱动。这个指针指向设备驱动中的一个函数，它在物理网络上传送一个包，当一个包被发送时它由IP层调用。这个字段由设备驱动初始化函数填充。output函数的第三个参数ipaddr是主机的IP地址，它可以接收实际链路层的帧。它不应和IP包的目的地址相同。特别地，当发送一个IP包到不在本地网络的主机时，链路层帧被发送到网络上一个路由。这种情况，给output函数的IP地址将是路由的IP地址。最后，state指针指向设备驱动中网络接口的特定状态，由设备驱动设置[30]。

LWIP仅仅实现IP最基本的功能，它可以发送，接收和转发包，但不能发送或接收分割的IP包，也不能处理带IP选项的包。对于大多数应用来说这不会引起任何问题。

使用由TCP/IP协议栈提供的服务有二种方式；一种是直接调用在TCP 和UDP 模块中的函数，另一种就是使用LWIP API。TCP 和UDP 模块提供一个网络服务的基本接口。该接口基于回调，因此使用它的应用程序可能因此不必以连续方式进行操作。这使应用程序的编程更加困难并且应用代码更难理解。为了接受数据，应用程序登记一个协议栈的回调函数。回调函数同一个特定的连接联系在一起，当该连接的包到达时，回调函数被协议栈调用。

与TCP 和UDP 模块直接接口地应用程序，至少部份地保留在像TCP/IP协议栈这样的处理过程中。这归结于回调函数无法横跨处理界限调用的事实。这既有好处也有不足。好处是应用程序和TCP/IP 协议是在同一个处理过程中，发送和接收包时不用上下文切换。主要不足是，在任何长的连续计算过程中应用程序无法介入自己，因为TCP/IP处理无法与计算平行发生，因而丧失通讯性能。通过把应用程序分成两部分可以克服这一缺点，一部分应付通信一部分应付计算。负责通讯的部分驻留在TCP/IP 过程中，负责计算的部份将是一个单独的过程。TCP/IP的处理不应该与其他运算并行处理，这样将会降低通讯的性能，所以我们把应用程序分解为两个部分，一部分专注于处理通讯，另一部分做其他的运算。通讯的部分将包含在TCP/IP进层中，而其他的繁杂运算则作为一个独立的进程。

作为高级别的BSD socket API，它是不适宜用于一个最小限度的TCP/IP执行的.特别BSD SOCKETS要求在TCP/IP协议栈中将要发送的数据从应用程序拷贝到内部缓存。需要拷贝数据的原因是通常TCP/IP协议栈和应用程序一般都处在不同的保护领域。大多数时候应用程序是位于户进程而TCP/IP却在操作系统内核中。通过避免这额外的拷贝就可以大幅度的提高API的性能。同样地，这样的拷贝需要分配额外的内存，每个信息包都浪费了双倍的内存。

LWIP API是专为LWIP设计并利用LWIP的内部结构达成效果，LWIP API 和BSD API非常类似，但操作相对低级。LWIP API不需要TCP/IP和应用程序之间的

相互拷贝数据，应用程序可以巧妙的直接处理内部缓存。由于BSD SOCKET API很容易理解且已经有很多人为它写过应用程序，LWIP API很有必要有与BSD SOCKET 的兼容层面。

LWIP的函数处理流程图，如图5-23所示：

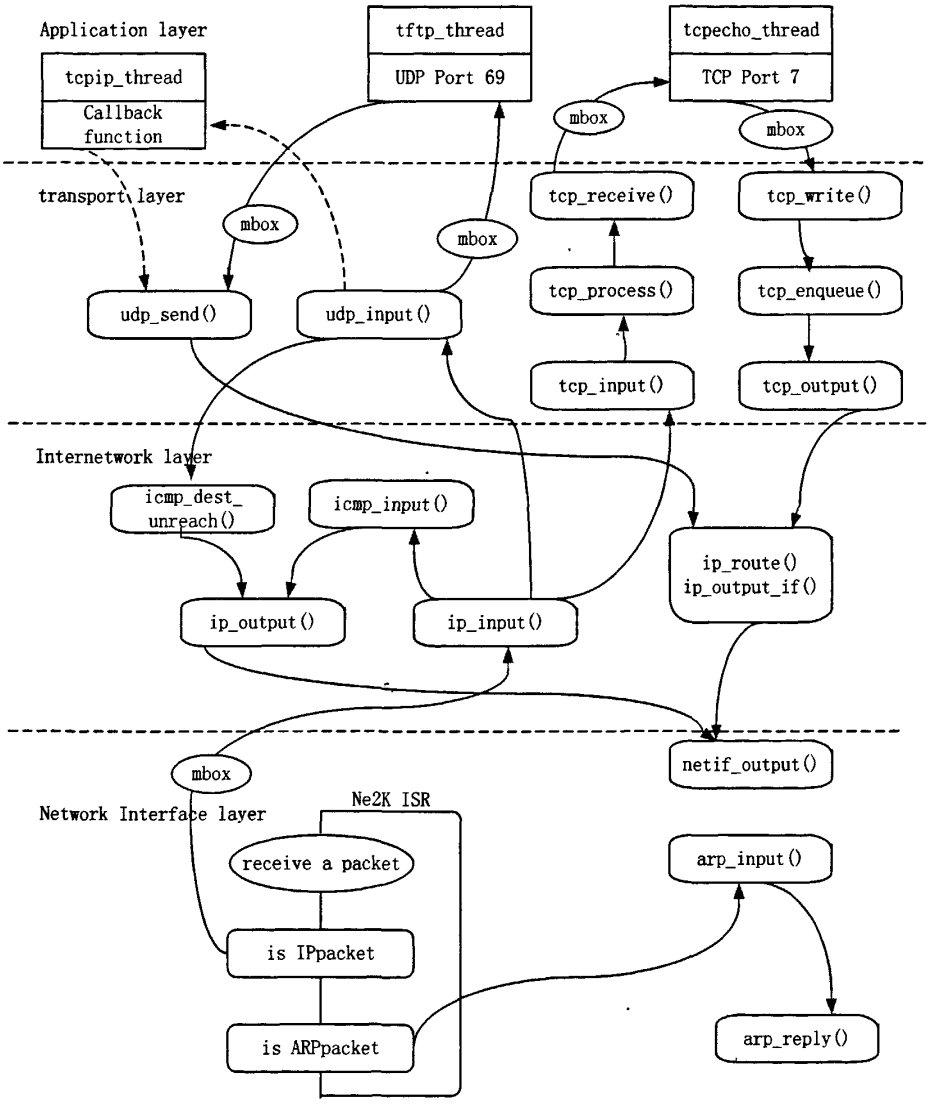

图5-23 LWIP的函数处理流程图

从应用程序去看，BSD SOCKET API的数据处理都是在一个连续的内存区域完成的。这是因为应用程序通常也是在这样的一个连续大内存块中完成数据处理

的。LWIP采用这样的机制是没有优势的，因为IWIP通常处理的数据缓存都被分割成了小的内存块。在通过应用程序前这些数据就会不得不要复制到一个连续的内存区，这将浪费双方的处理时间和内存，因此LWIP API允许应用程序巧妙地直接处理分离的缓存去避免额外的拷贝。

LWIP API 尽管非常类似 BSD SOCKET API，可是却有着值得注意的不同的地方，应用程序使用BSD SOCKET API 时候不需要知道普通文件和网络连接之间的差别，但使用LWIP API的时候就必须要知道确实在使用网络连接。网络数据被接收到分离的内存块的时候是以缓存的形式出现的，由于很多应用程序都希望在一个连续的内存区域处理数据，这就要有个函数去把这些缓存碎片复制到连续的内存空间中。发送出去的网络数据根据是TCP连接还是UDP是不同地处理的。在TCP连接时，数据通过一个指向连续内存区的指针发送，TCP/IP协议为传送区分适当大小的数据包和数据队列。在发送UDP数据的时候，应用程序将明确地分配缓存并填充数据，在输出函数被调用的时候由TCP/IP协议栈马上发送出去。

API是分做两个部分执行的,在图5-24中我们可以看到，一部分位于在TCP/IP的进程模块中,一部分当作连接库在应用程序中执行，这两部分的API通过由操作系统仿真层提供的进程间通讯(IPC)传达信息.当前使用的IPC机制有以下3种：

1.共享内存

2.消息

3.信号

当操作系统支持这些IPC类型的时候，并不意味着他它们得到了操作系统的最底层的支持，因为操作系统并不是一开始就支持它们,只是操作系统的的仿真层仿真了它们。

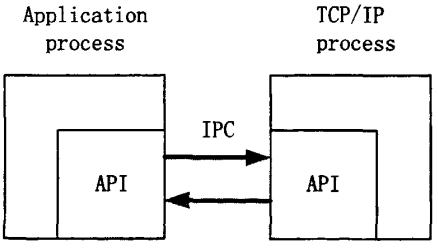

图5-24 分开为两部分的API执行

一般的设计原理都是尽可能地提高TCP/IP进程里的API的工作能力更胜于应用程序里API.这很重要，因为大部分的进程都用TCP/IP进程进行它们的TCP/IP通讯。遵循API部分的代码足迹，和应用程序连接的这部分并不是很重要.这些代码

可以在进程间共享,即使操作系统并不支持共享连接库。这些代码还可以保存在ROM中，嵌入系统一般尽管处理能力不高却拥有很大的ROM空间。

缓存管理器位于API执行库里，buffer在应用程序进程中被建立，复制和分配。应用程序和TCP/IP进程间使用共享内存来传递buffer，应用程序中用于通讯的buffer数据类型是一种提取于pbuf的类型。buffer传输引用到的内存，和分配的内存不同，它是可以利用共享内存的。所以可以进程间共享引用的内存，运行LWIP的嵌入式操作系统一般有意不做任何形式的内存保护，所以不会有问题。操作网络连接的函数位于TCP/IP进程的API中，应用程序的API函数会通过一个简单的通讯协议传递一个信息个TCP/IP进程的API，这信息包含操作的类型和操作的相关变量。这个操作由TCP/IP的API传输后用信息给应用程序一个返回值。

5.2.3 OS20 操作系统介绍

STLite/OS20（以下简称OS20）嵌入式实时操作系统内核提供了完善的多任务服务，多任务之间的同步和通信可以通过信号灯（Semaphore）和消息队列（Message queues）来实现。事件处理以中断的方式进行，并通过信号灯与任务通信。任务所需内存的分配，既可以通过操作系统来管理，也可以由用户自己管理。任务可以被分配不同的优先级，并根据优先级进行调度，同时提供时间函数，用于实现时间计算、延时等操作。

实时操作系统是所有上层程序代码运行的基础，OS20 支持所有的ST20 处理器内核，并对不同型号的ST20 内核做了重新优化，充分利用了芯片的特性，为基于ST20 的嵌入式系统开发，提供了高效优质的多任务环境。

OS20 操作系统内核具有如下特点：

- 基于16 个优先级的多优先级任务（Task）调度；
- 信号灯（Semaphores）；
- 消息队列（Message queue）；
- 时间管理器（Timer）；
- 存储器管理（Memory Management）；
- 中断处理（Interrupt）。

OS20 内核是整个操作系统的核心，它控制着程序中所有的进程，实现基于优先级的多任务调度。所以在系统开始运行前将内核初始化，OS20 初始化通常是主函数Main()中的第一个操作。

任务是实时操作系统的主要组成部分，多任务之间由内核根据某种调度算法分时执行，表现为并发执行。任务描述了应用程序某个离散、独立或部分行为，

任务之间可以相互通信。当程序刚开始执行的时候，只有一个主任务运行，其他任务在执行过程中被分别创建。各任务在内存中有自己的数据区，包括任务的堆栈和当前状态。这些数据区可以由OS20 在系统分区中动态分配或由用户静态分配。代码、全局静态数据区和堆栈区可以在任务之间共享，两个任务可以使用相同的代码而不会相互影响[31]。

ST20 CPU 内核里有两个时钟寄存器，即高优先级时钟寄存器和低优先级时钟寄存器。在高优先级时钟计数超过了一个设定值后，CPU 就认为一个时间片周期结束了。当一个任务（低优先级进程）的连续执行期内有两个时间片周期结束，处理器会将该任务调到任务表的尾部，开始执行其它任务。如果一个低优先级任务被高优先级的任务剥夺了控制权，那么当低优先级的任务恢复时，它的时间片周期将继续从时间片周期的开始处计时。STx5105 采用的处理器内核是ST20 系列CPU，它的硬件支持高和低两个优先级。运行于低优先级的任务的优先级别可以由用户定义，每个任务的初始优先级是在任务创建时定义的。但是根任务（root task）的优先级不用设置，它固定为最高优先级的任务。OS20 系统共支持16 个优先级[32]。

信号灯的主要功能是在多个任务之间提供一种简单有效的同步。在OS20中，规定了两种不同类型的请求信号灯的任务队列：

1、正常情况下，任务的队列是根据申请信号灯的先后次序决定，也就是FIFO 型。这时创建以FIFO 为基础的信号灯。

2、若希望高优先级的任务能早些获得信号灯，可以创建以优先级为基础的信号灯。

OS20 提供三种使用信号灯的方法：

1、同时允许多个任务来访问一个共享资源，允许的最大个数在该信号灯产生时就已经确定。如果有超过规定个数的任务要访问资源，那它必须等到占有资源的任务释放了信号灯才可以访问。

2、不论在什么时候只允许一个任务访问共享资源，这对于相互排斥地访问同一块共享资源是很有用的。在这种情况下，信号灯的令牌数应该初始化为1。

3、信号灯用作同步时，通常是进行任务和中断处理程序之间的同步。这时信号灯初始化为1，任务会执行信号灯等待命令。中断产生后执行中断服务程序，在中断服务程序中释放信号灯，等待的任务就会继续执行。

通常一个任务在等待信号灯时，只要没有获得信号灯，这个任务就会永远等待下去。但是OS20 还提供了一种超时（timeout）机制，可以在申请信号灯时定义申请的最长等待时间，如果在这个时间内任务没有获得信号灯，任务会继续原来的程序。系统提供的有关信号灯操作函数有：

OSAL_SemInit( )/OSAL_SemCreate( )：用于生成新的信号灯变量；

OSAL_SemDelete( )：删除已生成的信号灯变量；

OSAL_SemWait( )/ OSAL_SemRelease( )：等待/释放信号灯变量的操作[33]。

消息队列用于为任务之间提供数据通信。在OS20 中，消息队列的实现涉及两种类型，一种是目前还没有使用的消息缓冲区——空闲队列；另一种是已经发送但还没有收到的消息，所以还占用着消息缓冲区——发送队列。所有消息缓冲区均在这两个队列之间循环。在使用消息队列时，首先要创建该队列，然后从空闲队列中获得一个缓冲区，最后才可以进行消息的发送和接收。使用完毕一定要将消息队列删除，否则会耗尽系统资源。在消息的使用上同样也可以利用超时机制。系统对消息队列的处理是在创建任务时指定该任务相关消息的数量和一个消息的字节长度，用户发送的消息由指定的任务接收。系统提供的消息管理函数包括：

OSAL_MsgSend( )：发送消息到指定的任务；

OSAL_MsgSendWithTimeOut( )：发送消息到指定的任务，超时则丢弃；

OSAL_MsgWait( )：等待接收消息；

OSAL_MsgCapabilityGet( )：获取消息队列中空闲的消息数量。

对于实时系统而言时钟是至关重要的，时钟以Tick 为单位，对于STx5105，不论CPU 工作频率是多少，一秒钟都为15625Ticks。在OS20 中，系统提供了一系列基本函数来管理时钟：

time_minus( )：两个时钟相减；

time_now( )：返回当前时钟；

time_plus( )：两个时钟相加。

系统还为软件定时器的使用提供了通用的接口函数，给定时器的使用带来了方便。控制软件定时器的函数有：

OSAL_TimerInit( )/ OSALi_TimerTerminate( )：初始化/结束定时器；

OSAL_TimerCreate( ) / OSAL_TimerDelete( )：创建/删除定时器；

OSAL_TimerConfigure( )：配置一个已创建的定时器；

OSAL_TimerStart( )/ OSAL_TimerStop( )：开始/停止一个配置好的定时器；

OSAL_TimerStateGet( )：获取指定定时器的状态[34]。

OS20 提供了很方便的内存管理函数。动态内存管理，内存块的分配和释放是在内存分区中进行的。内存分区的内容主要包括：管理内存块、被分配和释放的内存状态以及用于分配的算法。STLite/OS20 支持三种不同的内存分区方式：堆（heap）、固定（Fixed）和简单（simple）方式。不同的内存分区方式允许在执行时间与内存使用之间做出选择。本课题中的内存管理方式采用的是堆

方式，以malloc 和free 的形式来管理内存。系统提供的主要管理内存的函数有：

OSAL_MemoryAlloc( )：分配内存；

OSAL_MemoryFree( )：释放内存，系统对释放的空间进行回收；

OSAL_MemoryRealloc( )：重新分配内存。

OS20 提供了一套完整的中断处理函数，以便使紧急事件能够获得CPU 的控制权。通常只要有中断发生，CPU 将立即停止执行当前的任务，转而执行该中断处理程序，该过程全部由硬件完成。中断的优先级在中断初始化中进行设置。

STx5105 的中断电路由中断控制器和中断级别控制器组成，中断级别控制器最多可以支持64 个内部中断源和8 个外部中断源以及16 个中断级别，不同的中断源通过它对应到不同的中断级别上，然后由中断控制器响应处理中断。中断经常是由驱动层所控制的，常用到的中断管理函数有：

OSAL_IntEnable( )/OSAL_IntDisable( )：使能或禁止某中断；

OSAL_IntLock( )/OSAL_IntUnlock( )：使能或禁止所有的中断[35]。

5.2.4 移植需要注意的问题

LWIP的process model是所有TCP/IP协议栈都在一个进程当中，这样TCP/IP协议栈就和操作系统内核分开了，而应用层程序既可以是单独的进程也可以驻留在TCP/IP进程中，如果应用程序是单独的进程可以通过操作系统的邮箱，消息队列等和TCP/IP进程进行通讯。驱动层也可以驻留在TCP/IP进程中，通过LWIP提供的内部回调函数来实现。需要注意与操作系统，应用层和驱动层的封装问题。

1.与操作系统的封装

LWIP为了适应不同的操作系统，在代码中没有使用和某一个操作系统相关的系统调用和数据结构，而是在LWIP和操作系统之间增加了一个操作系统封装层。操作系统封装层为操作系统服务(定时、进程同步、消息传递)提供了一个统一的接口。在LWIP中使用semaphore实现同步和消息传递采用mbox(在OS20中我们使用的是Message Queue来实现LWIP中的“mbox”)。

独立出来的操作系统封装层的原代码在sys.c文件和一些结构定义在sys_arch.h头文件中，与操作系统相关的结构和函数主要分为:信号量、消息队列、创建新线程和定时器。

(1)信号量

LWIP协议栈之间的通信以及与应用层通信需要应用到信号量。因此需要在sys_arch.h中实现信号量结构体sys_sem_t，在sys.c文件中实现信号量处理函数

sys_sem_new( )，sys_sem_free( )，sys_sem_signal( )和sys_arch_sem_wait( )，在OS20系统中已经覆盖到信号量，能实现对信号量的各种操作，并且功能和上面几个函数类似，利用系统中函数对上面几个函数进行封装即可实现。

(2)消息队列

LWIP中消息队列用来实现数据报文的缓冲和传递，因此要实现消息队列中消息结构体sys_mbox_t和相对应的一系列操作函数sys_mbox_new( )，sys_mbox_free( )，sys_mbox_post( )，sys_arch_mbox_fetch( )。

利用OS20中的对消息结构体的定义和实现消息队列相应操作的函数，封装LWIP要求实现的消息队列结构体和具体操作函数，使LWIP能实现消息承载的数据在应用层、TCP/IP协议栈和低层驱动进行传递和管理。对消息队列的管理需要在LWIP实现，而LWIP消息队列实现需要借助OS20的函数，在内存管理模块实现对消息的创建、使用和删除。两部分综合起来完成LWIP的消息队列功能。

(3)创建新线程

在OS20中并没有LWIP中关于线程的概念，它是通过创建任务和任务管理来实现线程，因此可以把OS20对应的创建任务的函数进行封装，实现LWIP中创建新线程函数sys_thread_new( )。需要注意的是，LWIP的线程中并没有OS20任务的优先级的概念，涉及到创建多个线程时要按照具体情况分配好相应优先级别，OS20任务管理在有优先级别的情况下是抢占式的，优先级别相同时按时间片方式处理。

(4)定时器

LWIP中每个和TCP/IP相关的任务的一系列的定时事件组成一个单向链表，每个链表的起始指针存在lwip_timeouts的对应表项中。创建一个任务来控制所有定时事件，通过LWIP定时的单向链表实现对所有定时事件的管理，提高了效率。

2.与应用层的封装

应用程序的接口可直接由LWIP提供的API例如BSDsocket来实现。API的实现包括两个部分：一是提供API函数来实现与应用程序的连接，二是API的处理在TCP/IP协议栈和底层驱动来实现。两部分之间使用一个线程来处理，处理模式采用进程间通讯(IPC)机制进行通信。实现IPC机制有三种方式：共享内存、消息传递和信号量(semaphore)。基本的设计原则是尽可能多的工作在应用层完成，少部分通过API在TCP/IP处理中完成，数据处理上尽量避免物理拷贝时间。运行在应用处理上API和运行在TCP/IP处理上的API之间使用共享内存的方式来实现数据之间的传递。在TCP/IP处理中的API执行的操作和返回通过消息传递到应用处理中，消息应该包括被执行的操作类型和相应参数。

3.与驱动层的封装

LWIP在处理协议层和网络接口层上，使两者之间保持独立,将网卡驱动和协议层分开。在LWIP中支持多个网络接口，每个网络接口对应一个网卡结构体struct netif，这个netif结构包含了相应网络接口的属性、收发函数。网络接口的驱动主要用来实现网卡的初始化和中断处理以及网卡与以太网之间收和发。网络接口驱动与上层协议之间收发数据时，将直接调用netif结构提供的函数netif->input( )和netif->output( )，这样使得加载多个网卡驱动非常方便。

第六章 总结

本论文对于如何在数字电视机顶盒的嵌入式系统中实现和Internet的连接,实现电视网络和Internet的初步融合以及数据之间交流提出了解决方案和具体的实现步骤。将电视和Internet完美地结合起来，彻底改变了传统电视节目的单向传输方式，使观众和电视之间形成了一种互动的和谐关系，改变了过去人们被动地接受信息的状况，使人们可以主动的选取信息。

针对核心处理器为单片机的情况，本论文具体实现了TCP/IP协议栈的移植。实现了ARP、ICMP、TCP、UDP协议。ARP中实现了缓存（学习、更新、老化、轮转替换），并且上位机能够实现ping下位机。单片机的处理能力有限，不能处理大量的数据，但是这个模型的实现为实现DVB-C机顶盒接入Internet做了非常重要的铺垫，为在STx5105移植LWIP协议栈打下了重要的基础。对于DVB-C机顶盒接入Internet，本系统采用ST公司推出的基于STx5105 的方案，通过集成网卡芯片DM9000A，并在OS20操作系统上面移植LWIP协议栈，实现有线电视机顶盒与Internet互联的功能。本论文除了实现了TCP/IP协议栈的功能外，详细论述了LWIP协议栈的支持模块如：操作系统模拟层、缓冲和存储管理子系统、网络接口函数和一些处理因特网校验和的函数的具体实现。

移植TCP/IP协议栈到嵌入式系统中，使嵌入式系统可以连接Internet，已经成为嵌入式系统今后发展的一个重要方向，嵌入式系统能通过Internet发送有用的信息到远程终端上，为嵌入式设备实现远程数据采集、远程帮助、远程级等功能提供了可能，这是未来嵌入式系统发展的趋势。

参考文献

[1]杨建华，数字电视原理与应用，北京：北京航空航天大学出版社，2006.

[2]姜秀华、张永辉等，数字电视原理与应用，北京：人民邮电出版社，2003.8

[3]刘修文，数字电视有线传输技术，北京：电子工业出版社，2002.

[4]陈善广，计算机网络原理与通信技术，北京：清华大学出版社，2007

[5]杨延双,张建标,王全民，TCP/IP 协议分析及应用，北京：机械工业出版社，2007

[6]马争鸣，TCP/IP 原理与应用，北京：冶金工业出版社，2006

[7]兰少华，杨余旺，吕建勇，TCP/IP 网络与协议，北京：清华大学出版社，2006

[8]萧文龙，林松儒，TCP/IP 最佳入门，北京：机械工业出版社，2006

[9]王罡，林立志，基于 Windows 的 TCP/IP 编程，北京：清华大学出版社，2002

[10]Held G 著，戴志涛，郑岩译，以太网，第三版，北京：人民邮电出版社，1999

[11]Larry L Peterson，Bruce S Davie，Computer networks，北京：机械工业出版社，2001

[12]Specification of RTL8019AS (Realtek full duplex Ethernet controller with plug and play function).Realtekse2mi2conductorco.Ltd, URL: www.realtek.com.tw

[13]Wolf W 著，孙玉芳译，嵌入式计算系统设计原理，北京：机械工业出版社，2002

[14]探矽工作室，嵌入式系统开发圣经，北京：中国青年出版社，2002.

[15]许海燕，付炎，嵌入式系统技术与应用，北京：机械工业出版社，2002

[16]谭浩强，C 程序设计，北京：清华大学出版社，2001

[17]Keil Software Corp., Keil C51 uVision2 Getting Started User's Guide, Feb.2001

[18]余永权，嵌入式系统、智能家电及家居网络，单片机与嵌入式系统应用，2001

[19]于红光，Visual Basic 程序设计教程，上海：上海交通大学出版社，2006

[20]周必水，Visual Basic 程序设计实践教程，北京：科学出版社，2004

[21]姚巍，Visual Basic数据库开发从入门到精通，北京：人民邮电出版社，

2006

[22]鲁士文，认识和使用TCP/IP，北京：电子工业出版社，1998

[23]张曾科，计算机网络，北京：清华大学出版社，2005

[24]马海军，吴华，TCP/IP协议原理与应用，北京：清华大学出版社，2005

[25]王勇，姚亦峰，陈抗生，一种嵌入式系统接入 Internet 的方法及实现[J]，电子技术，2000

[26]STx5105 Datasheet, September 2005

[27]老古，单片机与TCP/IP网络，老古论坛网，2002

[28]DUNKELS A. Design and Imp lementation of the LWIP TCP / IP Stack[EB /OL]. http: / /www.sics.se /～adam, 2001

[29]Michael Barr 著，于志宏译，C/C++嵌入式系统编程(美)，北京：中国电力出版社，2001

[30]Gregory Sator & Doug Brown 著，张铭泽译，C++语言核心，北京：中国电力出版社，2001

[31]STMicroelectronics Corp., OS20 User Manual, Nov.2004.

[32]5105SingFTA_ALPHA REFERENCE SOFTWARE, October 2005.

[33] OS20 Real Time Operating System, Nov. 2003.

[34]张炜，基于 STLite/OS20 嵌入式操作系统的数字化处理电视的实现：[硕士学位论文]，天津；天津大学，2005

[35]STLite/OS20 Real-Time Kenrel Refeernce Manual, SM icroelectronics, 1998

发表论文和参加科研情况说明

发表的论文：

[1] 马永涛，刘开华，高新立，吕西午，“基于 GENE8310 的嵌入式多媒体网络通信系统设计”，《电子测量技术》，2007 年 3 月

参与的科研项目：

本人在攻读硕士学位期间，有幸加入天津市数字信息技术研究中心，参与了两个项目的研发工作：

1. 负责 TCP/IP 协议栈的研究工作和到嵌入式系统的移植工作

2. 本人参与过数字电视软件收发端软件的研发工作，负责数字电视机顶盒 Ipanel 中间件的移植工作，改写过接口函数，进一步熟悉了 OS20 操作系统和 makefile 函数

致　谢

本研究及学位论文是在我的导师刘开华教授的亲切关怀和悉心指导下完成的。刘开华教授严肃的科学态度，严谨的治学精神，精益求精的工作作风和科学的工作方法深深地感染和激励着我。从课题的选择到项目的最终完成，刘教授都始终给予我细心的指导和不懈的支持。两年多来，刘教授不仅在学业上给我以精心指导，同时还在思想、生活上给我以无微不至的关怀，在此谨向刘教授致以诚挚的谢意和崇高的敬意。

李燕青老师对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。在实验室工作及撰写论文期间，张耀丹、吴鹏、陈磊、罗砚、周静等同学对我论文中的 TCP/IP 研究工作给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢我的家人和一直在帮助我支持我的男朋友，他们的理解和支持使我能够在学校专心完成我的学业，在这里给他们致以最衷心的感谢。

在论文即将完成之际，我的心情无法平静，从开始进入课题到论文的顺利完成，有多少可敬的师长、同学、朋友给了我无言的帮助，在这里请接受我诚挚的谢意！