

深入剖析LwIP中的各种协议和应用。

以LwIP源码为核心，结合经典的云-管-端物联网应用实例，由浅入深，讲解LwIP技术和应用开发。配套野火STM32 M4/M7系列开发板，提供完整源代码，极具操作性。



0110.0110.0110.0110

# LwIP应用开发 实战指南

## 基于STM32

刘火良 杨森 编著

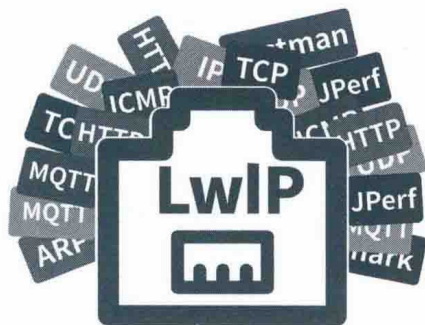


机械工业出版社  
China Machine Press



野火

野火嵌入式系列



0110.0110.0110.0110

# LwIP应用开发 实战指南



## 基于STM32

刘火良 杨森 编著



机械工业出版社  
China Machine Press

## 图书在版编目 (CIP) 数据

LwIP 应用开发实战指南：基于 STM32 / 刘火良, 杨森编著. —北京：机械工业出版社, 2019.9

(电子与嵌入式系统设计丛书)

ISBN 978-7-111-63582-6

I. L… II. ①刘… ②杨… III. 微控制器—系统开发—指南 IV. TP332.3-62

中国版本图书馆 CIP 数据核字 (2019) 第 188830 号

# LwIP 应用开发实战指南：基于 STM32

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：赵亮宇

责任校对：李秋荣

印刷：三河市宏图印务有限公司

版次：2019 年 9 月第 1 版第 1 次印刷

开本：186mm × 240mm 1/16

印张：28.75

书号：ISBN 978-7-111-63582-6

定价：119.00 元

客服电话：(010) 88361066 88379833 68326294

投稿热线：(010) 88379604

华章网站：www.hzbook.com

读者信箱：hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

## 作者简介

**刘火良** 野火电子创始人，嵌入式软件工程师，长期从事STM32软硬件开发，网络教程“零死角玩转STM32”的作者，《STM32库开发实战指南》的合著者。

**杨森** 野火电子合伙人，嵌入式软件工程师，专注于STM32和周边软件开发。合著有《STM32库开发实战指南》，已累计印刷十余次，广受STM32开发者的喜爱。



## 阅 读 建 议

全书内容循序渐进，不断迭代，前一章是后一章的基础，尽量不要跳跃性阅读，从理解原理到源码的实现是个很重要也很枯燥的过程，一定要耐心学习。此外，在学习的时候务必做到两点：一是不能一味地看书，要把代码和书结合起来学习，一边看书一边调试代码；二是在学完每一章之后，将配套例程重写一遍，做到举一反三，真正理解。

本书电子版和配套的源码均可到野火论坛（[www.fireBBS.cn](http://www.fireBBS.cn)）下载，欢迎各位读者积极参与交流分享。

# 前 言

## 如何学习本书

本书围绕 LwIP 2.1.2 版本源码讲解 TCP/IP 网络协议栈的基本知识，带领读者进入网络的世界。无论你是学生、嵌入式开发者还是物联网开发者，都可以从本书中学习到网络的相关知识，了解网络协议栈的处理思想。

本书将深入分析网络协议栈的原理与实现过程，涉及 ARP、IP、ICMP、TCP、UDP、HTTP、MQTT 等协议，还将深入讲解 LwIP 中内存管理、pbuf 数据包、网卡接口管理的原理与实现，并详细介绍 LwIP 的移植过程，读者可以将其移植到无操作系统/有操作系统的环境中使用。

除此之外，本书还通过理论和实践相结合，使用 LwIP 接入当前较大的几个云平台，如阿里云、百度云、OneNET，以使读者熟练掌握接入方法。在本书中，读者可以熟练掌握 LwIP 中 Netconn API 与 Socket API 的使用方式，并且掌握多种网络调试工具，如 Wireshark、Postman 以及 MQTT.fx 等的使用方法。

全书内容循序渐进，不断迭代，建议读者在学习的时候做到两点：一是不能一味看书，要把代码和理论结合起来学习，一边看书，一边调试代码。通过执行每一个程序，检验程序的执行流程和执行效果与自己想的是否一致，不断总结经验；二是在每学完一章之后，必须将配套的例程重写一遍（切记不要复制，即使是一个符号），举一反三，确保真正理解。

## 推荐阅读

- LwIP 官方源代码
- 《STM32 库开发实战指南》(已由机械工业出版社出版多个版本)
- 《FreeRTOS 内核实现与应用开发实战指南：基于 STM32》(已由机械工业出版社出版，ISBN 978-7-111-61825-6)

## 本书的技术论坛

如果在学习过程中遇到问题，可以到野火电子论坛（[www.firebbs.cn](http://www.firebbs.cn)）发帖交流，开源共享，共同进步。

鉴于水平有限，本书难免有错漏之处，热心的读者也可把勘误发送到论坛以便我们加以改进。祝你学习愉快，LwIP 的世界，野火与你同行！

# 目 录

前 言	
第 1 章 网络协议概述	1
1.1 常用网络协议	1
1.2 网络协议的分层模型	2
1.3 协议层报文间的封装与拆封	3
第 2 章 LwIP 概述	5
2.1 LwIP 的优缺点	5
2.2 LwIP 的文件说明	6
2.2.1 获取 LwIP 源码文件	6
2.2.2 LwIP 文件说明	8
2.3 LwIP 的说明文档	11
2.4 使用 vs code 查看源码	14
2.4.1 查看文件中的符号列表 和函数列表	14
2.4.2 函数定义跳转	15
2.5 LwIP 源码里的示例	16
2.6 LwIP 的 3 种编程接口	17
2.6.1 RAW/Callback API	17
2.6.2 Netconn API	18
2.6.3 Socket API	19
第 3 章 开发平台	20
3.1 以太网概述	20
3.1.1 PHY 层	20
3.1.2 MAC 子层	21
3.2 STM32 的 ETH 外设	23
3.3 MII 和 RMII 接口	24
3.4 PHY: LAN8720A	25
3.5 硬件设计	27
3.6 软件设计	29
3.6.1 获取 STM32 的裸机工程 模板	29
3.6.2 添加 bsp_eth.c 与 bsp_ eth.h	29
3.6.3 修改 stm32f4xx_hal_conf.h 文件	35
第 4 章 LwIP 的网络接口管理	37
4.1 netif 结构体	37
4.2 netif 的使用	41
4.3 与 netif 相关的底层函数	45
4.4 ethernetif.c 文件内容	46
4.4.1 ethernetif 数据结构	46
4.4.2 ethernetif_init()	47
4.4.3 low_level_init()	48
第 5 章 LwIP 的内存管理	50
5.1 几种内存分配策略	50

5.1.1	固定大小的内存块	50	6.7.3	pbuf_take()、pbuf_copy()、 pbuf_chain() 和 pbuf_ref()	88
5.1.2	可变长度分配	51	6.8	网卡中使用的 pbuf	88
5.2	动态内存池	52	6.8.1	low_level_output()	88
5.2.1	内存池的预处理	52	6.8.2	low_level_input()	91
5.2.2	内存池的初始化	57	6.8.3	ethernetif_input()	93
5.2.3	内存分配	58	<b>第 7 章 无操作系统移植 LwIP</b>	<b>94</b>	
5.2.4	内存释放	59	7.1	将 LwIP 添加到裸机工程	94
5.3	动态内存堆	61	7.2	移植头文件	97
5.3.1	内存堆的组织结构	61	7.3	移植网卡驱动	103
5.3.2	内存堆初始化	62	7.4	LwIP 时基	111
5.3.3	内存分配	64	7.5	协议栈初始化	112
5.3.4	内存释放	67	7.6	获取数据包	114
5.4	使用 C 库的 malloc 和 free 函数 来管理内存	71	7.6.1	查询方式	114
5.5	LwIP 中的配置	72	7.6.2	ping 命令详解	115
<b>第 6 章 网络数据包</b>	<b>74</b>		7.6.3	中断方式	116
6.1	TCP/IP 的分层思想	74	<b>第 8 章 有操作系统移植 LwIP</b>	<b>119</b>	
6.2	LwIP 的线程模型	75	8.1	向 LwIP 中添加操作系统	119
6.3	pbuf 结构体说明	76	8.1.1	复制 FreeRTOS 源码到 工程文件夹	119
6.4	pbuf 的类型	77	8.1.2	添加 FreeRTOS 源码到 工程组文件夹	120
6.4.1	PBUF_RAM 类型的 pbuf	78	8.1.3	指定 FreeRTOS 头文件 的路径	120
6.4.2	PBUF_POOL 类型的 pbuf	78	8.1.4	修改 stm32f10x_it.c	121
6.4.3	PBUF_ROM 和 PBUF_ REF 类型的 pbuf	80	8.2	lwipopts.h 文件需要加入的 配置	122
6.5	pbuf_alloc()	81	8.3	sys_arch.c/h 文件的编写	126
6.6	pbuf_free()	84	8.4	网卡底层的编写	136
6.7	其他 pbuf 操作函数	88	8.5	协议栈初始化	138
6.7.1	pbuf_realloc()	88			
6.7.2	pbuf_header()	88			

8.6 移植后使用 ping 命令测试基本 响应 .....	141	10.11.1 etharp_output() .....	182
<b>第 9 章 LwIP 一探究竟</b> .....	142	10.11.2 etharp_output_to_arp_ index() .....	185
9.1 网卡数据传入 LwIP 内核的流程 ..	142	10.11.3 etharp_query() .....	186
9.2 内核超时处理 .....	142	<b>第 11 章 IP</b> .....	193
9.2.1 sys_timeo 结构体与超时 链表 .....	144	11.1 IP 地址 .....	193
9.2.2 注册超时事件 .....	144	11.1.1 概述 .....	193
9.2.3 超时检查 .....	147	11.1.2 IP 地址编址 .....	193
9.3 tcpip_thread 线程 .....	149	11.1.3 特殊 IP 地址 .....	195
9.4 LwIP 中的消息 .....	151	11.2 局域网和广域网的概念 .....	196
9.4.1 消息结构 .....	151	11.2.1 局域网 .....	196
9.4.2 数据包消息 .....	153	11.2.2 广域网 .....	196
9.4.3 API 消息 .....	154	11.3 网络地址转换 .....	197
9.5 揭开 LwIP 的神秘面纱 .....	158	11.4 IP 数据报 .....	198
<b>第 10 章 ARP</b> .....	159	11.5 IP 数据报的数据结构 .....	202
10.1 数据链路层概述 .....	159	11.6 IP 数据报分片 .....	204
10.2 MAC 地址的基本概念 .....	159	11.7 IP 数据报发送 .....	208
10.3 初识 ARP .....	160	11.8 IP 数据报接收 .....	214
10.4 以太网帧结构 .....	160	<b>第 12 章 ICMP</b> .....	222
10.5 IP 地址映射为物理地址 .....	161	12.1 ICMP 功能简介 .....	222
10.6 ARP 缓存表 .....	162	12.2 ICMP 报文结构 .....	223
10.7 ARP 缓存表的超时处理 .....	165	12.3 ICMP 报文类型 .....	224
10.8 ARP 报文 .....	167	12.3.1 ICMP 差错报告报文 ..	224
10.9 发送 ARP 请求包 .....	170	12.3.2 ICMP 查询报文 .....	226
10.10 数据包接收流程 .....	172	12.4 LwIP 中的 ICMP 实现 .....	227
10.10.1 以太网中数据包的 接收 .....	172	12.4.1 ICMP 报文数据结构 ..	227
10.10.2 ARP 数据包处理 .....	176	12.4.2 发送 ICMP 差错报文 ..	229
10.10.3 更新 ARP 缓存表 .....	178	12.4.3 处理 ICMP 报文 .....	231
10.11 数据包发送流程 .....	182	<b>第 13 章 TCP</b> .....	235
		13.1 TCP 服务概述 .....	235

13.2	TCP 的特性	235	14.3	UDP 报文	266
13.2.1	连接机制	235	14.4	UDP 报文的数据结构	267
13.2.2	确认与重传	235	14.4.1	UDP 报文首部结构体	267
13.2.3	缓冲机制	236	14.4.2	UDP 控制块	267
13.2.4	全双工通信	236	14.5	UDP 报文发送	270
13.2.5	流量控制	236	14.6	UDP 报文接收	272
13.2.6	差错控制	237			
13.2.7	拥塞控制	237			
13.3	端口号的概念	237	<b>第 15 章 使用 Netconn 接口</b>		
13.4	TCP 报文段结构	239	编程		
13.4.1	TCP 报文段的封装	239	15.1	netbuf 结构体	277
13.4.2	TCP 报文段格式	239	15.2	netbuf 相关函数说明	279
13.5	TCP 连接	243	15.2.1	netbuf_new()	279
13.5.1	“三次握手”建立连接	243	15.2.2	netbuf_delete()	279
13.5.2	“四次挥手”终止连接	246	15.2.3	netbuf_alloc()	280
13.6	TCP 状态	247	15.2.4	netbuf_free()	280
13.6.1	LwIP 中定义的 TCP 状态	247	15.2.5	netbuf_ref()	281
13.6.2	TCP 状态转移	248	15.2.6	netbuf_chain()	282
13.7	TCP 中的数据结构	250	15.2.7	netbuf_data()	282
13.8	窗口的概念	253	15.2.8	netbuf_next() 与 netbuf_first()	282
13.8.1	接收窗口	254	15.2.9	netbuf_copy()	283
13.8.2	发送窗口	254	15.2.10	netbuf_take()	284
13.9	TCP 报文段处理	255	15.2.11	其他操作 netbuf 的宏定义	285
13.9.1	报文段缓冲队列	255	15.3	netconn 结构体	285
13.9.2	TCP 报文段发送	256	15.4	netconn 函数接口说明	287
13.9.3	TCP 报文段接收	260	15.4.1	netconn_new()	287
			15.4.2	netconn_delete()	288
			15.4.3	netconn_getaddr()	290
			15.4.4	netconn_bind()	290
			15.4.5	netconn_connect()	291
			15.4.6	netconn_disconnect()	292
<b>第 14 章 UDP</b>					
14.1	UDP 概述	265			
14.2	UDP 常用端口号	266			

15.4.7	netconn_listen()	293	16.4	实验	329
15.4.8	netconn_accept()	293	16.4.1	TCP Client 实验	329
15.4.9	netconn_recv()	295	16.4.2	TCP Server 实验	331
15.4.10	netconn_send()	296	16.4.3	UDP 实验	333
15.4.11	netconn_sendto()	297			
15.4.12	netconn_write()	298	<b>第 17 章 使用 RAW API 接口编程</b>		<b>335</b>
15.4.13	netconn_close()	301	17.1	RAW API 的 UDP 编程	335
15.5	实验	302	17.1.1	新建控制块: udp_ new()	335
15.5.1	TCP Client 实验	302	17.1.2	绑定控制块: udp_ bind()	336
15.5.2	TCP Client 实验现象	307	17.1.3	建立会话: udp_ connect()	337
15.5.3	TCP Server 实验	308	17.1.4	断开会话: udp_ disconnect()	338
15.5.4	TCP Server 实验现象	314	17.1.5	接收数据: udp_ recv()	339
15.5.5	UDP 实验	315	17.1.6	发送数据: udp_send() 与 udp_sendto()	339
15.5.6	UDP 实验现象	319	17.1.7	删除 UDP 控制块: udp_remove()	341
<b>第 16 章 使用 Socket 接口编程</b>		<b>321</b>	17.2	RAW API 的 TCP 编程	341
16.1	什么是 Socket	321	17.2.1	新建控制块: tcp_ new()	341
16.2	LwIP 中的 Socket	321	17.2.2	绑定控制块: tcp_ bind()	342
16.3	Socket API	322	17.2.3	控制块监听: tcp_ listen()	343
16.3.1	socket()	322	17.2.4	处理连接: tcp_ accept()	345
16.3.2	bind()	323	17.2.5	建立连接: tcp_ connect()	346
16.3.3	connect()	324	17.2.6	终止连接: tcp_ close()	348
16.3.4	listen()	325			
16.3.5	accept()	325			
16.3.6	read()、recv()、 recvfrom()	325			
16.3.7	sendto()	326			
16.3.8	send()	327			
16.3.9	write()	327			
16.3.10	close()	327			
16.3.11	ioctl()、ioctlsocket()	327			
16.3.12	setsockopt()	328			
16.3.13	getsockopt()	329			

17.2.7	接收数据: tcp_recv() ...	350	19.3	HTTP 报文 .....	379
17.2.8	发送数据: tcp_sent() ...	350	19.4	使用 Postman 获取论坛数据 .....	380
17.2.9	异常处理: tcp_err() .....	351	19.5	使用开发板获取论坛数据 .....	382
17.2.10	周期性回调: tcp_poll() .....	351	<b>第 20 章</b>	<b>HTTP 服务器 .....</b>	<b>386</b>
17.2.11	构建报文段: tcp_write() .....	352	20.1	Hello World 网页 demo .....	386
17.2.12	更新接收窗口: tcp_recv() .....	352	20.2	提供网页控制 LED 开关的 功能 .....	389
17.3	实验 .....	353	<b>第 21 章</b>	<b>MQTT 协议 .....</b>	<b>394</b>
17.3.1	TCP Client 实验 .....	353	21.1	MQTT 协议概述 .....	394
17.3.2	TCP Server 实验 .....	358	21.2	MQTT 通信模型 .....	394
17.3.3	UDP 实验 .....	360	21.3	消息主题与服务质量 .....	395
<b>第 18 章</b>	<b>使用 JPerf 工具测试 网速 .....</b>	<b>362</b>	21.4	MQTT 控制报文 .....	396
18.1	iPerf 与 JPerf .....	362	21.4.1	固定报头 .....	396
18.2	测试网络速度 .....	362	21.4.2	可变报头 .....	397
18.2.1	获取 JPerf 网络测速 工具 .....	362	21.4.3	有效载荷 .....	400
18.2.2	测试开发板接收速度 (Netconn API) .....	363	21.5	移植 MQTT 协议 .....	401
18.2.3	测试开发板接收速度 (Socket API) .....	366	21.6	cJSON 移植 .....	407
18.2.4	测试开发板发送速度 (Netconn API) .....	369	<b>第 22 章</b>	<b>连接到百度天工物接入 .....</b>	<b>411</b>
18.2.5	测试开发板发送速度 (Socket API) .....	372	22.1	物接入概述 .....	411
18.3	提高 LwIP 网络传输的速度 .....	375	22.2	使用 IoT Hub .....	411
<b>第 19 章</b>	<b>HTTP .....</b>	<b>377</b>	22.2.1	创建计费套餐 .....	411
19.1	什么是 HTTP .....	377	22.2.2	创建项目 .....	411
19.2	URL 与资源 .....	378	22.2.3	创建策略 .....	413
			22.2.4	创建身份与创建 用户 .....	413
			22.2.5	MQTT 软件测试 连接 .....	415
			22.3	开发板连接 IoT Hub .....	417
			22.4	IoT Hub 的规则引擎 .....	422
			22.4.1	什么是规则引擎 .....	422

22.4.2 使用规则引擎 .....	422	23.4 阿里云物联网的规则引擎 .....	435
22.5 数据可视化 .....	424	<b>第 24 章 连接到 OneNET .....</b>	<b>437</b>
22.5.1 IoT Hub 的时序 数据库 .....	424	24.1 使用 OneNET .....	437
22.5.2 IoT Hub 的物可视 .....	426	24.2 测试连接 .....	438
<b>第 23 章 连接到阿里云物联 .....</b>	<b>429</b>	24.3 开发板连接 OneNET .....	439
23.1 使用阿里云物联 .....	429	24.4 添加数据流 .....	441
23.2 MQTT 软件测试连接 .....	431	24.5 系统主题的发布格式 .....	442
23.3 开发板连接阿里云物联 .....	434	24.6 使用开发板发布数据点 .....	443
		24.7 数据可视化 .....	446

# 第 1 章

## 网络协议概述

### 1.1 常用网络协议

互联网为人类社会带来巨大变革，几乎改变了人们生活的方方面面。互联网通信的本质是数字通信，任何数字通信都离不开通信协议，通信设备只有按照约定的、统一的方式去封装和解析信息，才能实现通信。互联网通信所要遵守的众多协议，被统称为 TCP/IP。

TCP/IP 是一个协议族，包含众多协议。对于网络应用开发人员，可能听到得更多的是其中的应用层协议，比如 HTTP（Hyper Text Transfer Protocol，超文本传输协议）、FTP（File Transfer Protocol，文件传输协议）、MQTT（Message Queuing Telemetry Transport，消息队列遥测传输）等。

HTTP 的应用最为广泛。比如大家日常使用计算机时的一个常规操作：打开计算机，再打开浏览器，输入网址，最后按下 Enter 键，这一刻你就开启了 HTTP 通信。HTTP 工作于 <客户端 - 服务端> 架构之上（服务端也称为服务器端，除特别说明外，本书出现的“服务端”即为“服务器端”），浏览器作为 HTTP 客户端，通过 URL 向 HTTP 服务端（即 Web 服务器）发送所有请求。Web 服务器根据接收到的请求向客户端发送响应信息。借助这种浏览器和服务器之间的 HTTP 通信，我们能够足不出户地获得来自世界各地的信息。另外，网页不仅仅是大型服务器的专利，在物联网风潮盛行的今天，许多随处可见的小型设备（如空调、冰箱、插座、路由器等）都内嵌网页，在物理链路畅通的情况下，用户可以用手机、平板电脑上的浏览器随时随地监控这些设备。

FTP 是工作在应用层的网络协议，使得主机间可以共享文件，用于在两台设备之间传输文件（双向传输）。它也是一个客户端 - 服务端框架系统。用户可以通过一个支持 FTP 的客户端程序，连接到远程主机上的 FTP 服务端程序，通过客户端程序向服务端程序发出命令，服务端程序执行用户所发出的命令，并将执行的结果返回客户端。FTP 除了支持基本的文件上传 / 下载功能外，还支持目录操作、权限设置、身份验证机制，许多网盘的文件传输功能都是基于 FTP 实现的。

在物联网发展的初期，物联网场景中的设备使用何种应用层协议进行通信一直是备受争议的话题。很多开发人员习惯了网页开发模式，于是经常选择 HTTP 作为通信方式。使用 HTTP 有以下不利因素：HTTP 是一种同步协议，设备需要等待服务器的响应才可以进行下

一步的工作，然而在设备数量多、网络不可靠的场景下，实现同步通信很困难；HTTP 是单向的，设备只能主动向服务器发出数据，无法被动接收来自网络的数据，这不适用于实时控制的场合；HTTP 是有许多帧头和规则的重量级协议，在设备中实现需要耗费大量的系统资源。基于上述形势，MQTT 和 COAP 等轻量级、异步的通信协议便得到了物联网设备开发者的青睐，尤其是 MQTT。

MQTT 协议是 IBM 公司于 1990 年设计并推出的一款通信协议，于 2014 年正式成为一个 OASIS 开放标准。近年来，MQTT 的应用呈现爆炸性的增长势头，大有一统物联网的趋势。另外，MQTT 在物联网以外的其他领域也得到了广泛的应用，比如许多公司在制作手机 APP 时，会使用 MQTT 来实现消息推送、即时聊天等功能。

嵌入式设备接入互联网的需求越来越大，有以下几点原因：

1) 近些年，各种带网络接入功能的 MCU、SoC 层出不穷，开源轻量的 TCP/IP 协议栈日趋成熟和完善，云平台的市场越来越繁荣，这些因素大大降低了嵌入式设备的入网成本，也为许多资源受限的低端设备接入互联网提供了可能。

2) “物联网+”的风潮日渐盛行，设备能够被远程监控，这一点已经成为许多产品的技术要求。

3) 人们对于设备“智能性”的需求越来越高，当今热门的大数据、图像处理、语音识别、机器学习等功能都可以集成在云端，成为云平台能提供的服务。终端设备大多是计算、存储能力有限的设备，这些设备如果想要获取“智能”，最便捷的办法就是接入云平台，利用各项云服务。

互联网的基础就是 TCP/IP。TCP/IP 是一个非常复杂的协议族，即使我们能把它的设计思想和实现原理都解释清楚，你也不一定有时间和精力去学习它，所以本书的写作重点不在于对 TCP/IP 的解读，而在于对它的应用。另外，TCP/IP 的复杂性也决定了它使用起来没那么简单，即使我们只关注应用开发，也依然需要对它的许多概念和设计思想有所了解，才能编写出正确、高效、健壮的应用程序。

希望能借此书，让嵌入式开发工程师以浓厚的兴趣和清晰的视野，搭上物联网发展的快车。

### 1.2 网络协议的分层模型

TCP/IP 是众多网络协议的集合，包括 ARP、IP、ICMP、UDP、TCP、DNS、HTTP、FTP、MQTT 等。这些协议按照功能，可以划分为几个不同的层次，如图 1-1 所示。我们在 1.1 节中介绍的 HTTP、FTP、MQTT 隶属于应用层。那么 TCP/IP 为什么需要分层，分层的依据又是什么呢？

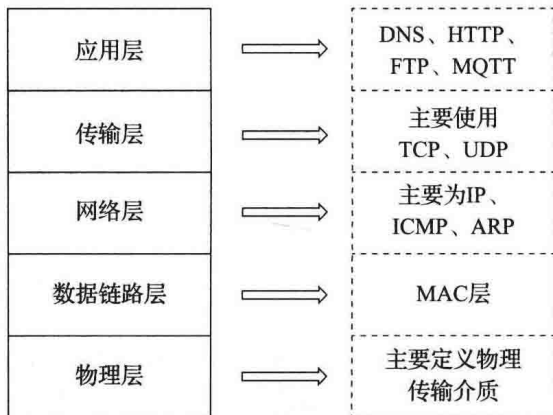


图 1-1 TCP/IP 的分层

TCP/IP 协议栈中不同协议所完成的功能是不一样的，某些协议的实现要依赖于其他协议，依据这种依赖关系，可以将协议栈分层。在图 1-1 中，低层协议为相邻的上层协议提供服务，是上层协议得以实现的基础。

其中，物理层（PHY）规定了传输信号所需要的物理电平、介质特征。数据链路层（MAC）规定了数据帧能被网卡接收的条件，最常见的方式是利用网卡的 MAC 地址，发送方会在欲发送的数据帧的首部加上接收方网卡的 MAC 地址信息，接收方只有监听到属于自己的 MAC 地址信息后，才会去接收并处理该数据。每台网络设备都应该有自己的网络地址，网络层规定了主机的网络地址该如何定义，以及如何在网络地址和 MAC 地址之间进行映射，即 ARP，还实现了数据包在主机之间的传递，而一台主机内部可能运行着多个网络程序。传输层可以区分数据包是属于哪一个应用程序的，可以说传输层实现了数据包端到端的传递。另外，数据包在传输过程中可能会出现丢包、乱序和重复的现象，网络层并没有提供应对这些错误的机制，而传输层可以解决这些问题，如 TCP。应用层以下的各层完成了数据的传递，应用层则决定了你如何应用和处理这些数据。之所以会有许多应用层协议，是因为互联网中传递的数据种类很多、差异很大、应用场景十分多样。

### 1.3 协议层报文间的封装与拆封

本书的后面章节会对 TCP/IP 协议栈中的每层协议进行分析和讲解。在这里，我们以图 1-2 简单解释一下在数据的发送和接收过程中，TCP/IP 都做了什么。

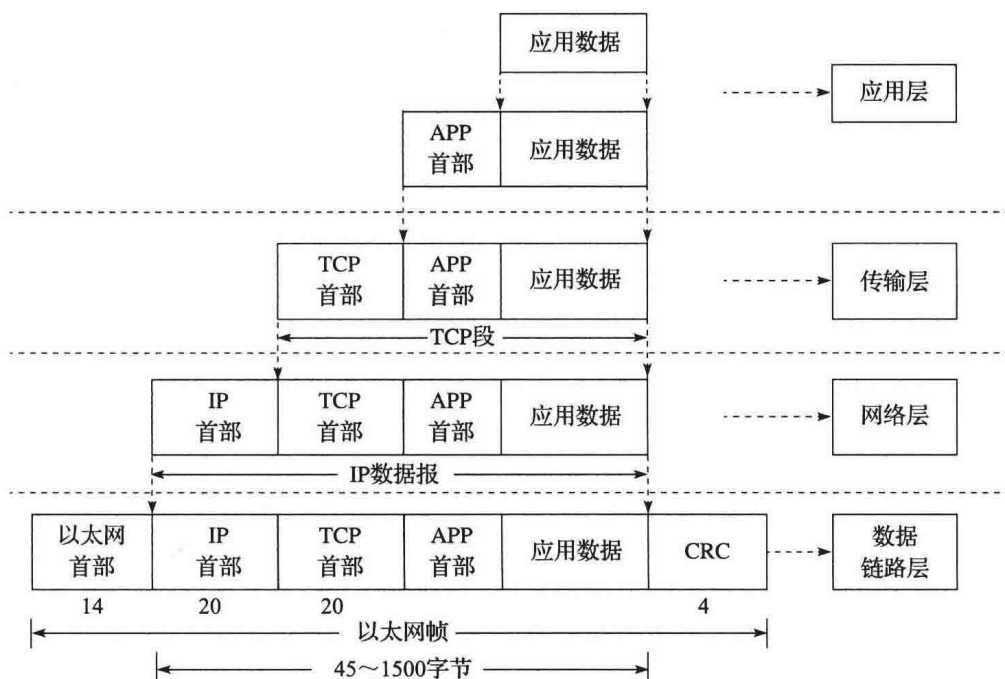


图 1-2 TCP/IP 协议栈各层的报文封装与拆封