

# 【新提醒】网络编程懒人入门 (一)：快速理解网络通信协议 (上篇) - 网络编程/专项技术 区 - 即时通讯开发者社区!



关注我的公众号

即时通讯技术之路，你并不孤单!

IM开发 / 实时通信 / 网络编程

原作者：阮一峰(ruanyifeng.com)，本文由即时通讯网重新整理发布，感谢原作者的无私分享。

## 1、写在前面

论坛和群里常会有技术同行打算自己开发IM或者消息推送系统，很多时候连基本的网络编程理论（如网络协议等）都不了解，就贸然定方案、写代码，显得非常盲目且充满技术风险。

即时通讯网论坛里精心整理了《[\[通俗易懂\]深入理解TCP协议](#)》、《[不为人知的网络编程](#)》、《[P2P技术详解](#)》、《[高性能网络编程](#)》这几个网络编程的系列文章，甚至还有图文并貌+实战代码的《[NIO框架入门](#)》等等。资料虽

好，无奈很多同行或许是时间紧迫，也或许是心态浮躁，反正就是没办法静下心来仔细研读，导致错过了很多必须掌握的网络编程知识基础（如果您正打算从零开发移动端IM，则建议您从此文开始《[新手入门一篇就够：从零开发移动端IM](#)》）。

本次《网络编程懒人入门》系列文章（共3篇），将为大家（尤其是上面说的浮躁的开发者同行）提供懒人快速入门，希望在你没办法耐心读完上面的几个系列文章（但还是强烈建议优先去读一读）的情况还能对基本的网络编程知识有所了解和掌握，从而对您的IM系统或消息推系统的技术选型、方案制定、代码编写起到理论支撑作用。

本文将从网络通信协议讲起，懒人们，动起来^^！

## 2、正文引言

我们每天使用互联网，你是否想过，它是如何实现的呢？

全世界几十亿台电脑，连接在一起，两两通信。上海的某一块网卡送出信号，洛杉矶的另一块网卡居然就收到了，两者实际上根本不知道对方的物理位置，你不觉得这是很神奇的事情吗？

互联网的核心是一系列协议，总称为"互联网协议"（Internet Protocol Suite）。它们对电脑如何连接和组网，做出了详尽的规定。理解了这些协议，就理解了互联网的原理。

下面就是我的学习笔记。因为这些协议实在太复杂、太庞大，我想整理一个简洁的框架，帮助自己从总体上把握它们。为了保证简单易懂，我做了大量的简化，有些地方并不全面和精确，但是应该能够说清楚互联网的原理。

另外，如果您很好奇承载这些网络协议的物理设备是怎么工作的，可以先看看《[网络编程懒人入门\(六\)：史上最通俗的集线器、交换机、路由器功能原理入门](#)》。

### 3、系列文章

本文是系列文章中的第1篇，本系列文章的大纲如下：

- 《[网络编程懒人入门\(一\)：快速理解网络通信协议\\_ \(上篇\)\\_](#)》（本文）
- 《[网络编程懒人入门\(二\)：快速理解网络通信协议\\_ \(下篇\)\\_](#)》
- 《[网络编程懒人入门\(三\)：快速理解TCP协议一篇就够](#)》
- 《[网络编程懒人入门\(四\)：快速理解TCP和UDP的差异](#)》
- 《[网络编程懒人入门\(五\)：快速理解为什么说UDP有时比TCP更有优势](#)》
- 《[网络编程懒人入门\(六\)：史上最通俗的集线器、交换机、路由器功能原理入门](#)》
- 《[网络编程懒人入门\(七\)：深入浅出，全面理解HTTP协议](#)》

- [《网络编程懒人入门\(八\): 手把手教你写基于TCP的Socket长连接》](#)
- [《网络编程懒人入门\(九\): 通俗讲解, 有了IP地址, 为何还要用MAC地址? 》](#)

本站的《脑残式网络编程入门》也适合入门学习, 本系列大纲如下:

- [《脑残式网络编程入门\(一\): 跟着动画来学TCP三次握手和四次挥手》](#)
- [《脑残式网络编程入门\(二\): 我们在读写Socket时, 究竟在读写什么? 》](#)
- [《脑残式网络编程入门\(三\): HTTP协议必知必会的一些知识》](#)
- [《脑残式网络编程入门\(四\): 快速理解HTTP/2的服务器推送\(Server Push\)》](#)

如果您觉得本系列文章过于基础, 您可直接阅读《不为人知的网络编程》系列文章, 该系列目录如下:

- [《不为人知的网络编程\(一\): 浅析TCP协议中的疑难杂症\(上篇\)》](#)
- [《不为人知的网络编程\(二\): 浅析TCP协议中的疑难杂症\(下篇\)》](#)
- [《不为人知的网络编程\(三\): 关闭TCP连接时为什么会TIME\\_WAIT、CLOSE\\_WAIT》](#)

- [《不为人知的网络编程\(四\)：深入研究分析TCP的异常关闭》](#)
- [《不为人知的网络编程\(五\)：UDP的连接性和负载均衡》](#)
- [《不为人知的网络编程\(六\)：深入地理解UDP协议并用好它》](#)
- [《不为人知的网络编程\(七\)：如何让不可靠的UDP变的可靠？》](#)
- [《不为人知的网络编程\(八\)：从数据传输层深度解密HTTP》](#)
- [《不为人知的网络编程\(九\)：理论联系实际，全方位深入理解DNS》](#)

关于移动端网络特性及优化手段的总结性文章请见：

- [《现代移动端网络短连接的优化手段总结：请求速度、弱网适应、安全保障》](#)
- [《移动端IM开发者必读\(一\)：通俗易懂，理解移动网络的“弱”和“慢”》](#)
- [《移动端IM开发者必读\(二\)：史上最全移动弱网络优化方法总结》](#)

## 4、参考资料

[《TCP/IP详解 - 第11章·UDP：用户数据报协议》](#)

[《TCP/IP详解 - 第17章·TCP：传输控制协议》](#)

[《TCP/IP详解 - 第18章·TCP连接的建立与终止》](#)

[《TCP/IP详解 - 第21章·TCP的超时与重传》](#)

[《通俗易懂-深入理解TCP协议（上）：理论基础》](#)

[《通俗易懂-深入理解TCP协议（下）：RTT、滑动窗口、拥塞处理》](#)

[《理论经典：TCP协议的3次握手与4次挥手过程详解》](#)

[《理论联系实际：Wireshark抓包分析TCP 3次握手、4次挥手过程》](#)

[《计算机网络通讯协议关系图（中文珍藏版）》](#)

[《高性能网络编程\(一\)：单台服务器并发TCP连接数到底可以有多少》](#)

[《高性能网络编程\(二\)：上一个10年，著名的C10K并发连接问题》](#)

[《高性能网络编程\(三\)：下一个10年，是时候考虑C10M并发问题了》](#)

[《高性能网络编程\(四\)：从C10K到C10M高性能网络应用的理论探索》](#)

[《简述传输层协议TCP和UDP的区别》](#)

[《为什么QQ用的是UDP协议而不是TCP协议？》](#)

[《移动端即时通讯协议选择：UDP还是TCP？》](#)

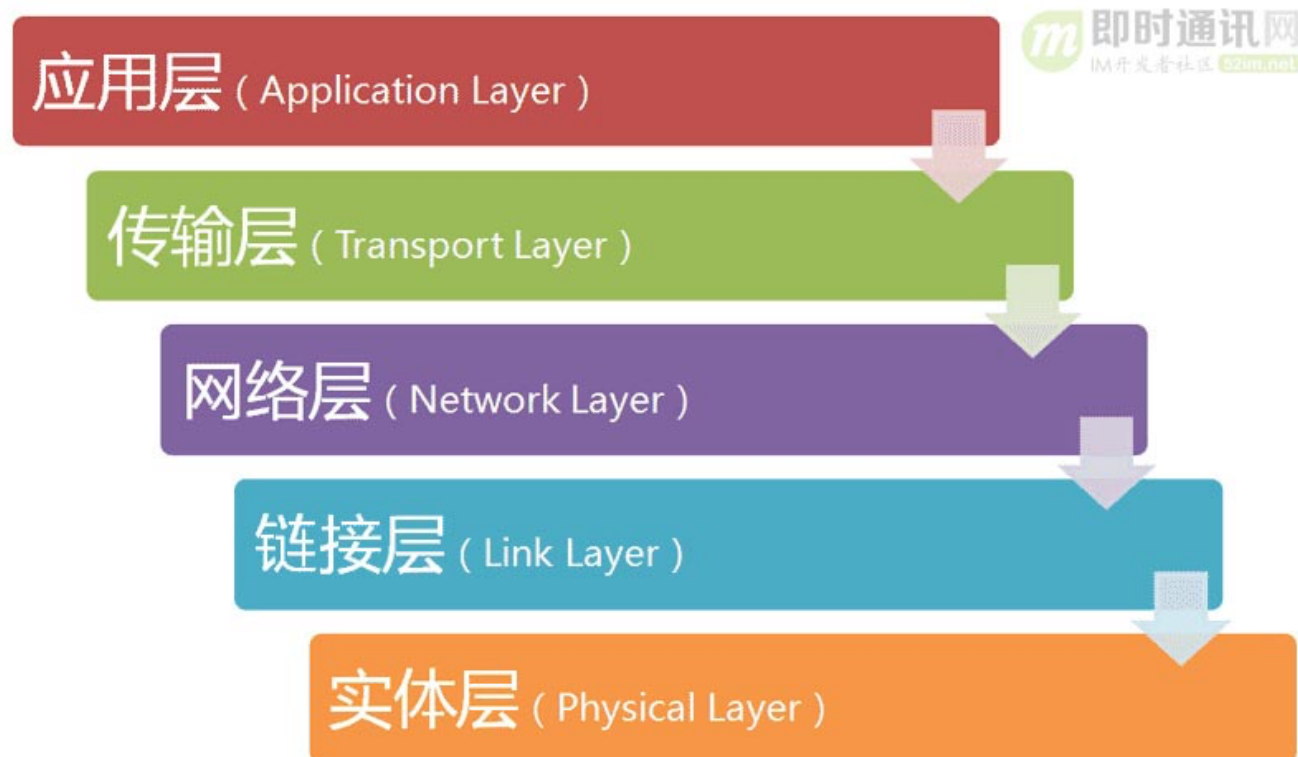
## 5、内容概述

### 5.1五层模型

互联网的实现，分成好几层。每一层都有自己的功能，就

像建筑物一样，每一层都靠下一层支持。用户接触到的，只是最上面的一层，根本没有感觉到下面的层。要理解互联网，必须从最下层开始，自下而上理解每一层的功能。

如何分层有不同的模型，有的模型分七层，有的分四层。我觉得，把互联网分成五层，比较容易解释：



如上图所示，最底下的一层叫做"实体层"（Physical Layer），最上面的一层叫做"应用层"（Application Layer），中间的三层（自下而上）分别是"链接层"（Link Layer）、"网络层"（Network Layer）和"传输层"（Transport Layer）。越下面的层，越靠近硬件；越上面的层，越靠近用户。

它们叫什么名字，其实并不重要。只需要知道，互联网分成若干层就可以了。

## 5.2层与协议

每一层都是为了完成一种功能。为了实现这些功能，就需要大家都遵守共同的规则。大家都遵守的规则，就叫做"协议"（protocol）。

互联网的每一层，都定义了很多协议。这些协议的总称，就叫做"互联网协议"（Internet Protocol Suite）。它们是互联网的核心，下面介绍每一层的功能，主要就是介绍每一层的主要协议。

## 6、实体层

我们从最底下的一层开始。

电脑要组网，第一件事要干什么？当然是先把电脑连起来，可以用光缆、电缆、双绞线、无线电波等方式。





这就叫做"实体层"，它就是把电脑连接起来的物理手段。它主要规定了网络的一些电气特性，作用是负责传送0和1的电信号。

## 7、链接层

### 7.1定义

单纯的0和1没有任何意义，必须规定解读方式：多少个电信号算一组？每个信号位有何意义？

这就是"链接层"的功能，它在"实体层"的上方，确定了0和1的分组方式。

### 7.2以太网协议

早期的时候，每家公司都有自己的电信号分组方式。逐渐地，一种叫做"以太网"（Ethernet）的协议，占据了主导地位。

以太网规定，一组电信号构成一个数据包，叫做"帧"（Frame）。每一帧分成两个部分：标头（Head）和数据（Data）。

"标头"包含数据包的一些说明项，比如发送者、接受者、数据类型等等；"数据"则是数据包的具体内容。

"标头"的长度，固定为18字节。"数据"的长度，最短为46字节，最长为1500字节。因此，整个"帧"最短为64字节，最长为1518字节。如果数据很长，就必须分割成多个帧进行发送。

## 7.3MAC地址

上面提到，以太网数据包的"标头"，包含了发送者和接受者的信息。那么，发送者和接受者是如何标识呢？

以太网规定，连入网络的所有设备，都必须具有"网卡"接口。数据包必须是从一块网卡，传送到另一块网卡。网卡的地址，就是数据包的发送地址和接收地址，这叫做MAC地址。



每块网卡出厂的时候，都有一个全世界独一无二的MAC地址，长度是48个二进制位，通常用12个十六进制数表示。

MAC Address

00-B0-D0-86-BB-F7

前6个十六进制数是厂商编号，后6个是该厂商的网卡流水号。有了MAC地址，就可以定位网卡和数据包的路径了。

## 7.4广播

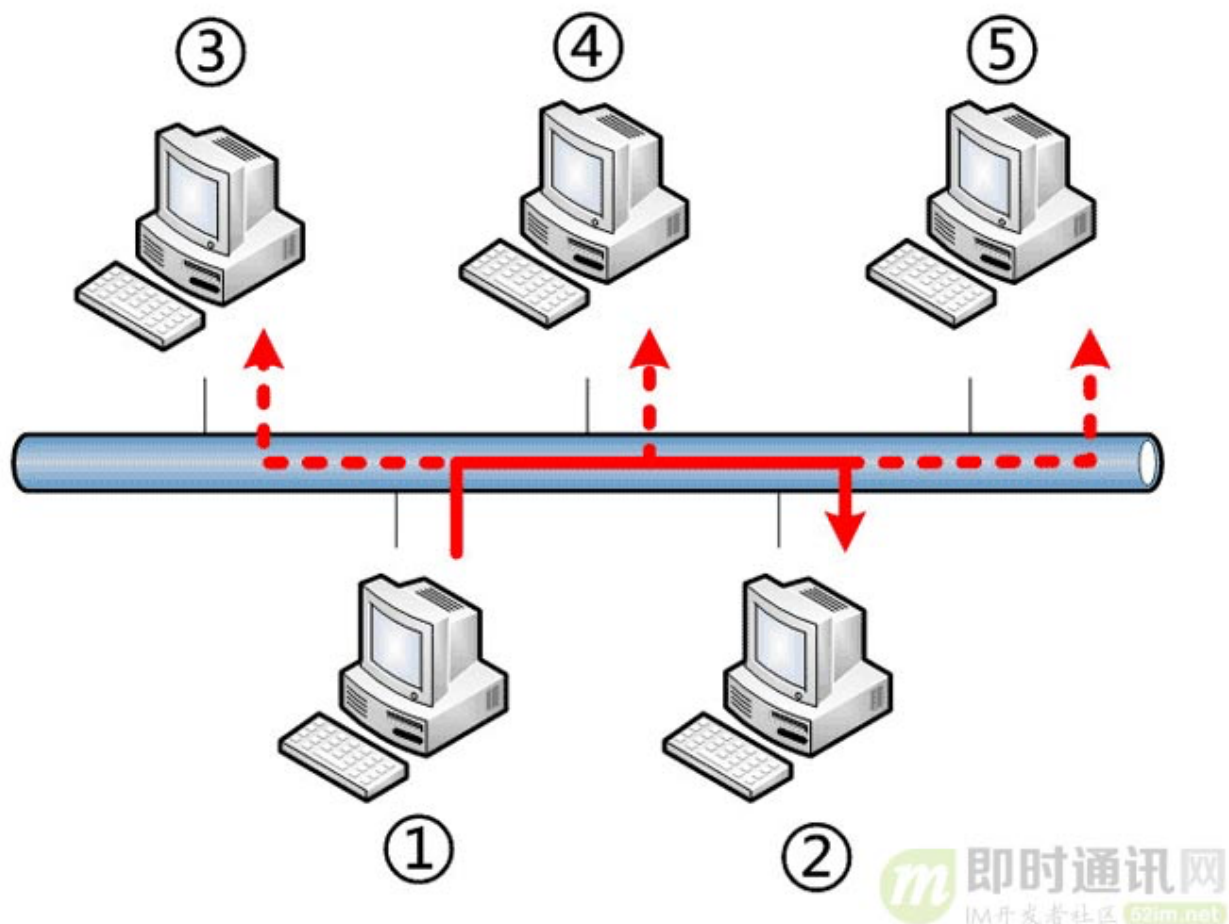
定义地址只是第一步，后面还有更多的步骤：

- **1) 首先：**一块网卡怎么会知道另一块网卡的MAC地址？

回答是有一种ARP协议，可以解决这个问题。这个留到后面介绍，这里只需要知道，以太网数据包必须知道接收方的MAC地址，然后才能发送。

- **2) 其次：**就算有了MAC地址，系统怎样才能把数据包准确送到接收方？

回答是以太网采用了一种很"原始"的方式，它不是把数据包准确送到接收方，而是向本网络内所有计算机发送，让每台计算机自己判断，是否为接收方。



上图中，1号计算机向2号计算机发送一个数据包，同一个子网络的3号、4号、5号计算机都会收到这个包。它们读取这个包的"标头"，找到接收方的MAC地址，然后与自身的MAC地址相比较，如果两者相同，就接受这个包，做进一步处理，否则就丢弃这个包。这种发送方式就叫做"广播"（broadcasting）。

有了数据包的定义、网卡的MAC地址、广播的发送方式，"链接层"就可以在台计算机之间传送数据了。

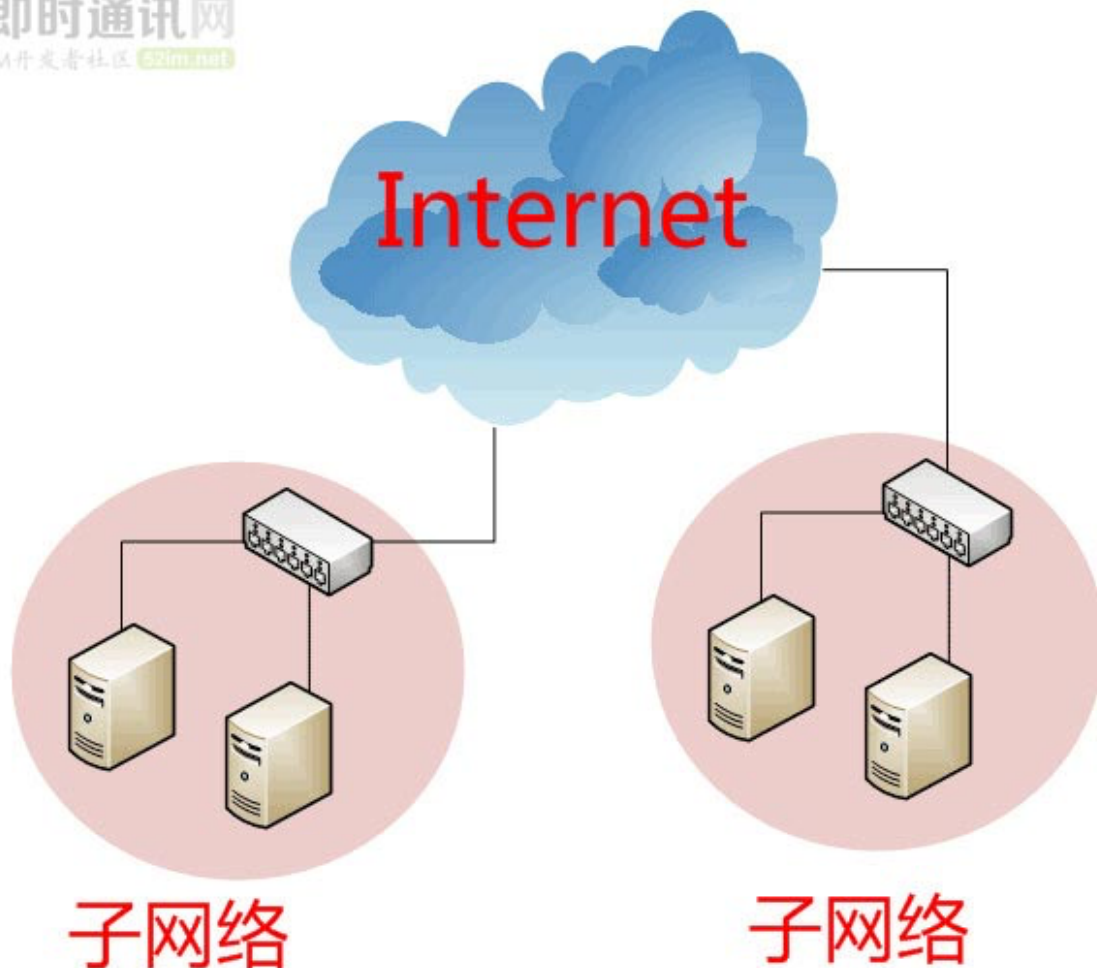
## 8、网络层

## 8.1网络层的由来

以太网协议，依靠MAC地址发送数据。理论上，单单依靠MAC地址，上海的网卡就可以找到洛杉矶的网卡了，技术上是可以实现。

但是，这样做有一个重大的缺点。以太网采用广播方式发送数据包，所有成员人手一"包"，不仅效率低，而且局限在发送者所在的子网络。也就是说，如果两台计算机不在同一个子网络，广播是传不过去的。这种设计是合理的，否则互联网上每一台计算机都会收到所有包，那会引起灾难。

互联网是无数子网络共同组成的一个巨型网络，很像想象上海和洛杉矶的电脑会在同一个子网络，这几乎是不可能的。



因此，必须找到一种方法，能够区分哪些MAC地址属于同一个子网络，哪些不是。如果是同一个子网络，就采用广播方式发送，否则就采用"路由"方式发送。（"路由"的意思，就是指如何向不同的子网络分发数据包，这是一个很大的主题，本文不涉及。）遗憾的是，MAC地址本身无法做到这一点。它只与厂商有关，与所处网络无关。

这就导致了"网络层"的诞生。它的作用是引进一套新的地址，使得我们能够区分不同的计算机是否属于同一个子网络。这套地址就叫做"网络地址"，简称"网址"。

于是，"网络层"出现以后，每台计算机有了两种地址，一种是MAC地址，另一种是网络地址。两种地址之间没有任

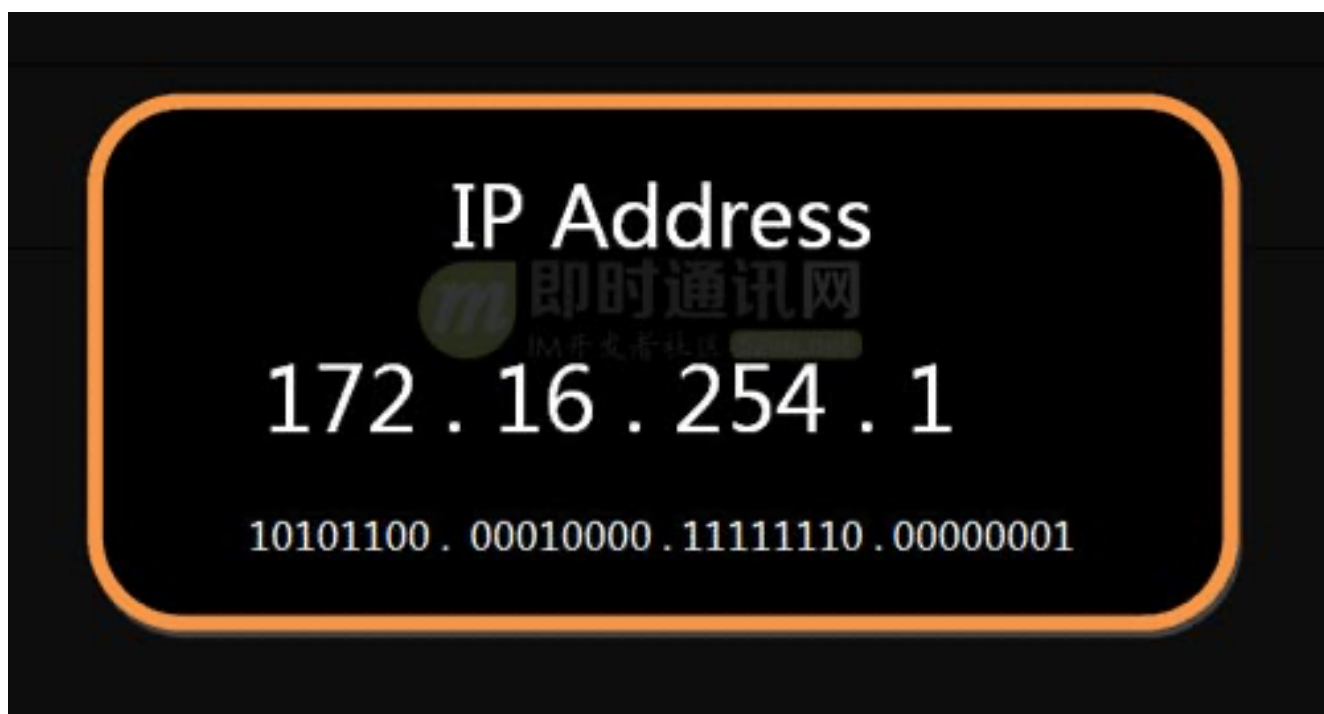
何联系，MAC地址是绑定在网卡上的，网络地址则是管理员分配的，它们只是随机组合在一起。

网络地址帮助我们确定计算机所在的子网络，MAC地址则将数据包送到该子网络中的目标网卡。因此，从逻辑上可以推断，必定是先处理网络地址，然后再处理MAC地址。

## 8.2IP协议

规定网络地址的协议，叫做IP协议。它所定义的地址，就被称为IP地址。目前，广泛采用的是IP协议第四版，简称IPv4。

**IPv4**这个版本规定，网络地址由**32**个二进制位组成：



习惯上，我们用分成四段的十进制数表示IP地址，从0.0.0.0一直到255.255.255.255。



互联网上的每一台计算机，都会分配到一个IP地址。这个地址分成两个部分，前一部分代表网络，后一部分代表主机。比如，IP地址172.16.254.1，这是一个32位的地址，假定它的网络部分是前24位（172.16.254），那么主机部分就是后8位（最后的那个1）。处于同一个子网络的电脑，它们IP地址的网络部分必定是相同的，也就是说172.16.254.2应该与172.16.254.1处在同一个子网络。

但是，问题在于单单从IP地址，我们无法判断网络部分。还是以172.16.254.1为例，它的网络部分，到底是前24位，还是前16位，甚至前28位，从IP地址上是看不出来的。

那么，怎样才能从IP地址，判断两台计算机是否属于同一个子网络呢？这就要用到另一个参数"子网掩码"（subnet mask）。

所谓"子网掩码"，就是表示子网络特征的一个参数。它在形式上等同于IP地址，也是一个32位二进制数字，它的网络部分全部为1，主机部分全部为0。比如，IP地址172.16.254.1，如果已知网络部分是前24位，主机部分是后8位，那么子网络掩码就是11111111.11111111.11111111.00000000，写成十进制就是255.255.255.0。

知道"子网掩码"，我们就能判断，任意两个IP地址是否处在同一个子网络。方法是将两个IP地址与子网掩码分别进行AND运算（两个数位都为1，运算结果为1，否则为

0) , 然后比较结果是否相同, 如果是的话, 就表明它们在同一个子网络中, 否则就不是。

比如, 已知IP地址172.16.254.1和172.16.254.233的子网掩码都是255.255.255.0, 请问它们是否在同一个子网络? 两者与子网掩码分别进行AND运算, 结果都是172.16.254.0, 因此它们在同一个子网络。

总结一下, IP协议的作用主要有两个, 一个是为每一台计算机分配IP地址, 另一个是确定哪些地址在同一个子网络。

## 8.3IP数据包

根据IP协议发送的数据, 就叫做IP数据包。不难想象, 其中必定包括IP地址信息。但是前面说过, 以太网数据包只包含MAC地址, 并没有IP地址的栏位。那么是否需要修改数据定义, 再添加一个栏位呢?

回答是不需要, 我们可以把IP数据包直接放进以太网数据包的"数据"部分, 因此完全不用修改以太网的规格。这就是互联网分层结构的好处: 上层的变动完全不涉及下层的结构。

具体来说, **IP数据包**也分为"标头"和"数据"两个部分:



"标头"部分主要包括版本、长度、IP地址等信息，"数据"部分则是IP数据包的具体内容。它放进以太网数据包后，以太网数据包就变成了下面这样：



IP数据包的"标头"部分的长度为20到60字节，整个数据包的总长度最大为65,535字节。因此，理论上，一个IP数据包的"数据"部分，最长为65,515字节。前面说过，以太网数据包的"数据"部分，最长只有1500字节。因此，如果IP数据包超过了1500字节，它就需要分割成几个以太网数据包，分开发送了。

## 8.4ARP协议

关于"网络层"，还有最后一点需要说明。因为IP数据包是放在以太网数据包里发送的，所以我们必须同时知道两个地址，一个是对方的MAC地址，另一个是对方的IP地址。

通常情况下，对方的IP地址是已知的（后文会解释），但是我们不知道它的MAC地址。

所以，我们需要一种机制，能够从IP地址得到MAC地址。

这里又可以分成两种情况：

- **1) 第一种情况：**如果两台主机不在同一个子网络，那么事实上没有办法得到对方的MAC地址，只能把数据包传送到两个子网络连接处的"网关"（gateway），让网关去处理；
- **2) 第二种情况：**如果两台主机在同一个子网络，那么我们可以用ARP协议，得到对方的MAC地址。ARP协议也是发出一个数据包（包含在以太网数据包中），其中包含它所要查询主机的IP地址，在对方的MAC地址这一栏，填的是FF:FF:FF:FF:FF:FF，表示这是一个"广播"地址。它所在子网络的每一台主机，都会收到这个数据包，从中取出IP地址，与自身的IP地址进行比较。如果两者相同，都做出回复，向对方报告自己的MAC地址，否则就丢弃这个包。

总之，有了ARP协议之后，我们就可以得到同一个子网络内的主机MAC地址，可以把数据包发送到任意一台主机之上了。

## 9、传输层

## 9.1传输层的由来

有了MAC地址和IP地址，我们已经可以在互联网上任意两台主机上建立通信。

接下来的问题是，同一台主机上有许多程序都需要用到网络，比如，你一边浏览网页，一边与朋友在线聊天。当一个数据包从互联网上发来的时候，你怎么知道，它是表示网页的内容，还是表示在线聊天的内容？

也就是说，我们还需要一个参数，表示这个数据包到底供哪个程序（进程）使用。这个参数就叫做"端口"（port），它其实是每一个使用网卡的程序的编号。每个数据包都发到主机的特定端口，所以不同的程序就能取到自己所需要的数据。

"端口"是0到65535之间的一个整数，正好16个二进制位。0到1023的端口被系统占用，用户只能选用大于1023的端口。不管是浏览网页还是在线聊天，应用程序会随机选用一个端口，然后与服务器的相应端口联系。

"传输层"的功能，就是建立"端口到端口"的通信。相比之下，"网络层"的功能是建立"主机到主机"的通信。只要确定主机和端口，我们就能实现程序之间的交流。因此，Unix系统就把主机+端口，叫做"套接字"（socket）。有了它，就可以进行网络应用程序开发了。

## 9.2UDP协议

现在，我们必须在数据包中加入端口信息，这就需要新的协议。最简单的实现叫做UDP协议，它的格式几乎就是在数据前面，加上端口号。

**UDP数据包**，也是由"标头"和"数据"两部分组成：



"标头"部分主要定义了发出端口和接收端口，"数据"部分就是具体的内容。然后，把整个UDP数据包放入IP数据包中的"数据"部分，而前面说过，IP数据包又是放在以太网数据包之中的，所以整个以太网数据包现在变成了下面这样：



UDP数据包非常简单，"标头"部分一共只有8个字节，总长度不超过65,535字节，正好放进一个IP数据包。

## 9.3 TCP协议

UDP协议的优点是比较简单，容易实现，但是缺点是可靠性较差，一旦数据包发出，无法知道对方是否收到。为了解决这个问题，提高网络可靠性，TCP协议就诞生了。这个协议非常复杂，但可以近似认为，它就是有确认机制的UDP协议，每发出一个数据包都要求确认。如果有一个数据包遗失，就收不到确认，发出方就知道有必要重发这个数据包了。

因此，TCP协议能够确保数据不会遗失。它的缺点是过程复杂、实现困难、消耗较多的资源。

TCP数据包和UDP数据包一样，都是内嵌在IP数据包的"数据"部分。TCP数据包没有长度限制，理论上可以无限长，但是为了保证网络的效率，通常TCP数据包的长度不会超过IP数据包的长度，以确保单个TCP数据包不必再分割。

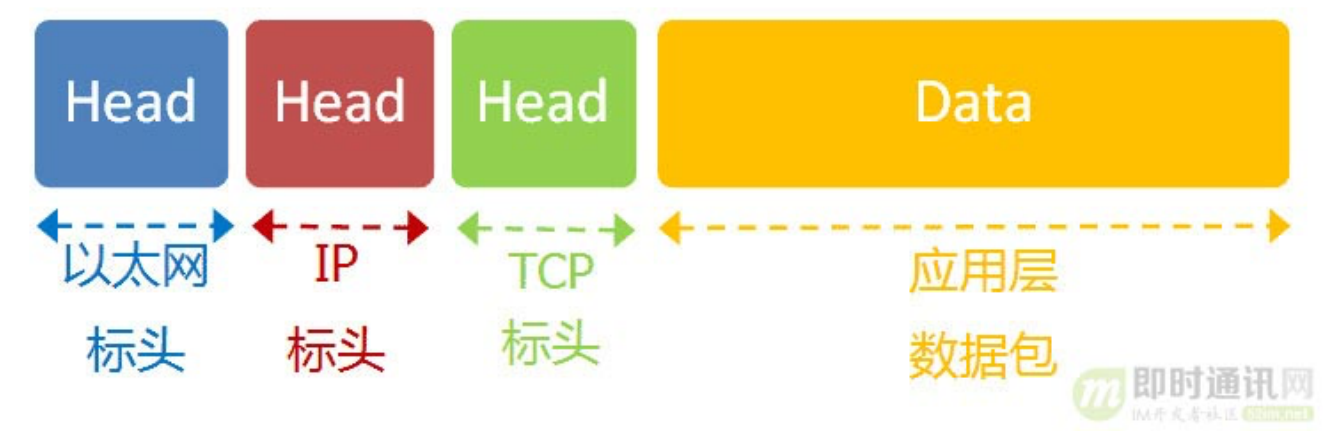
## 10、应用层

应用程序收到"传输层"的数据，接下来就要进行解读。由于互联网是开放架构，数据来源五花八门，必须事先规定好格式，否则根本无法解读。"应用层"的作用，就是规定应用程序的数据格式。

举例来说，TCP协议可以为各种各样的程序传递数据，比

如Email、WWW、FTP等等。那么，必须有不同协议规定电子邮件、网页、FTP数据的格式，这些应用程序协议就构成了"应用层"。这是最高的一层，直接面对用户。它的数据就放在TCP数据包的"数据"部分。

因此，现在的以太网的数据包就变成下面这样：



# 11、本文小结

至此，整个互联网的五层结构，自下而上全部讲完了。这是从系统的角度，解释互联网是如何构成的。下一篇《网络编程懒人入门(二)：快速理解网络通信协议（下篇）》，我反过来，从用户的角度，自上而下看看这个结构是如何发挥作用，完成一次网络数据交换的。敬请期待！

(原文链接：[点此进入](#)，有改动)

## 附录：更多网络编程资料



[《技术往事：改变世界的TCP/IP协议（珍贵多图、手机慎点）》](#)

[《UDP中一个包的大小最大能多大？》](#)

[《Java新一代网络编程模型AIO原理及Linux系统AIO介绍》](#)

[《NIO框架入门\(一\)：服务端基于Netty4的UDP双向通信Demo演示》](#)

[《NIO框架入门\(二\)：服务端基于MINA2的UDP双向通信Demo演示》](#)

[《NIO框架入门\(三\)：iOS与MINA2、Netty4的跨平台UDP双向通信实战》](#)

[《NIO框架入门\(四\)：Android与MINA2、Netty4的跨平台UDP双向通信实战》](#)

[《P2P技术详解\(一\)：NAT详解——详细原理、P2P简介》](#)

[《P2P技术详解\(二\)：P2P中的NAT穿越\(打洞\)方案详解》](#)

[《P2P技术详解\(三\)：P2P技术之STUN、TURN、ICE详解》](#)

[《通俗易懂：快速理解P2P技术中的NAT穿透原理》](#)

>> [更多同类文章 .....](#)