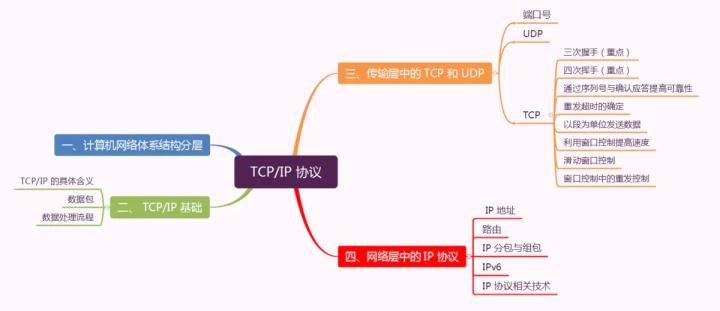
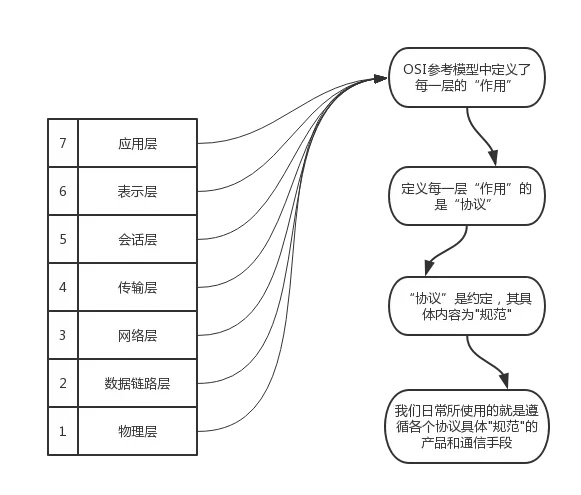
作者：iOSer  
链接：https://zhuanlan.zhihu.com/p/147370653  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**本文篇幅也比较长，先来一张思维导图，带大家过一遍。**



**一、 计算机网络体系结构分层**



**再者就是资源共享**

**作为一个开发者，有一个学习的氛围跟一个交流圈子特别重要，这是一个我的iOS交流群：**[**413038000**](https://link.zhihu.com/?target=https%3A//jq.qq.com/%3F_wv%3D1027%26k%3D5rQ90tC)**，不管你是大牛还是小白都欢迎入驻 ，分享BAT,阿里面试题、面试经验，讨论技术， 大家一起交流学习成长！**

**推荐阅读**

[**iOS开发——最新 BAT面试题合集（持续更新中）**](https://link.zhihu.com/?target=https%3A//www.jianshu.com/p/75e4b9fdcf41)

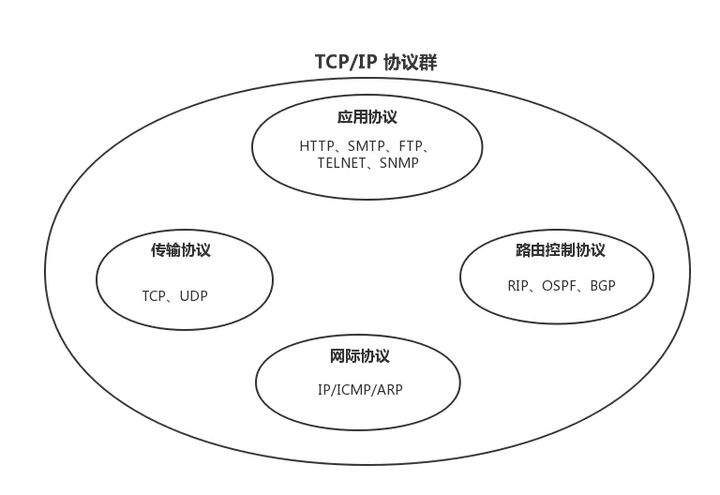


**不难看出，TCP/IP 与 OSI 在分层模块上稍有区别。OSI 参考模型注重“通信协议必要的功能是什么”，而 TCP/IP 则更强调“在计算机上实现协议应该开发哪种程序”。**

**二、 TCP/IP 基础**

**1. TCP/IP 的具体含义**

从字面意义上讲，有人可能会认为 TCP/IP 是指 TCP 和 IP 两种协议。实际生活当中有时也确实就是指这两种协议。然而在很多情况下，它只是利用 IP 进行通信时所必须用到的协议群的统称。具体来说，IP 或 ICMP、TCP 或 UDP、TELNET 或 FTP、以及 HTTP 等都属于 TCP/IP 协议。他们与 TCP 或 IP 的关系紧密，是互联网必不可少的组成部分。TCP/IP 一词泛指这些协议，因此，有时也称 TCP/IP 为网际协议群。  
互联网进行通信时，需要相应的网络协议，TCP/IP 原本就是为使用互联网而开发制定的协议族。因此，互联网的协议就是 TCP/IP，TCP/IP 就是互联网的协议。

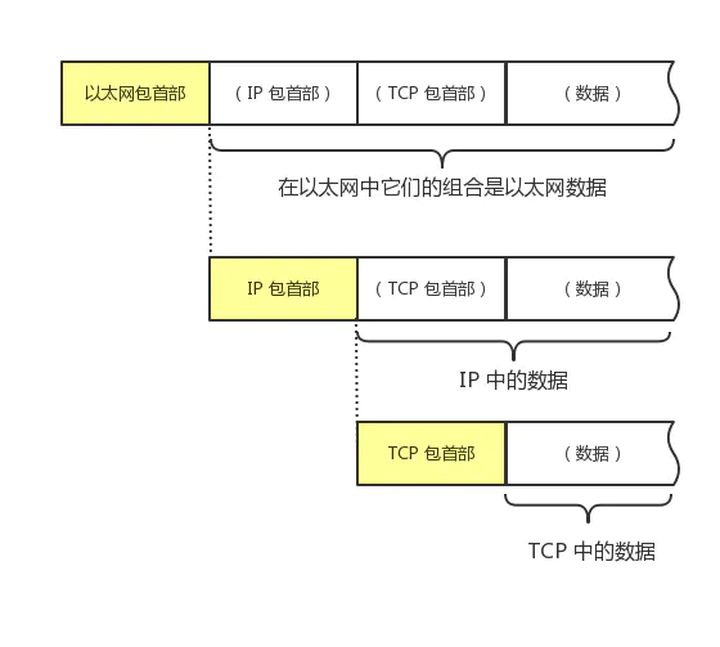


**2. 数据包**

**包、帧、数据包、段、消息**  
以上五个术语都用来表述数据的单位，大致区分如下：

* 包可以说是全能性术语；
* 帧用于表示数据链路层中包的单位；
* 数据包是 IP 和 UDP 等网络层以上的分层中包的单位；
* 段则表示 TCP 数据流中的信息；
* 消息是指应用协议中数据的单位。

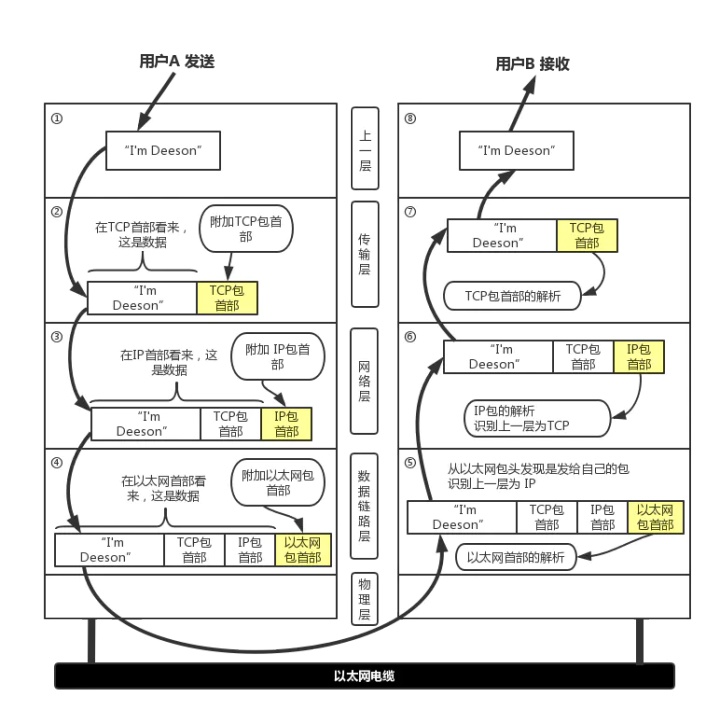
每个分层中，都会对所发送的数据附加一个首部，在这个首部中包含了该层必要的信息，如发送的目标地址以及协议相关信息。通常，为协议提供的信息为包首部，所要发送的内容为数据。在下一层的角度看，从上一层收到的包全部都被认为是本层的数据。



网络中传输的数据包由两部分组成：一部分是协议所要用到的首部，另一部分是上一层传过来的数据。首部的结构由协议的具体规范详细定义。在数据包的首部，明确标明了协议应该如何读取数据。反过来说，看到首部，也就能够了解该协议必要的信息以及所要处理的数据。**包首部就像协议的脸。**

**3. 数据处理流程**

下图以用户 a 向用户 b 发送邮件为例子：



* ① 应用程序处理  
  首先应用程序会进行编码处理，这些编码相当于 OSI 的表示层功能；  
  编码转化后，邮件不一定马上被发送出去，这种何时建立通信连接何时发送数据的管理功能，相当于 OSI 的会话层功能。
* ② TCP 模块的处理  
  TCP 根据应用的指示，负责建立连接、发送数据以及断开连接。TCP 提供将应用层发来的数据顺利发送至对端的可靠传输。为了实现这一功能，需要在应用层数据的前端附加一个 TCP 首部。
* ③ IP 模块的处理  
  IP 将 TCP 传过来的 TCP 首部和 TCP 数据合起来当做自己的数据，并在 TCP 首部的前端加上自己的 IP 首部。IP 包生成后，参考路由控制表决定接受此 IP 包的路由或主机。
* ④ 网络接口（以太网驱动）的处理  
  从 IP 传过来的 IP 包对于以太网来说就是数据。给这些数据附加上以太网首部并进行发送处理，生成的以太网数据包将通过物理层传输给接收端。
* ⑤ 网络接口（以太网驱动）的处理  
  主机收到以太网包后，首先从以太网包首部找到 MAC 地址判断是否为发送给自己的包，若不是则丢弃数据。  
  如果是发送给自己的包，则从以太网包首部中的类型确定数据类型，再传给相应的模块，如 IP、ARP 等。这里的例子则是 IP 。
* ⑥ IP 模块的处理  
  IP 模块接收到 数据后也做类似的处理。从包首部中判断此 IP 地址是否与自己的 IP 地址匹配，如果匹配则根据首部的协议类型将数据发送给对应的模块，如 TCP、UDP。这里的例子则是 TCP。  
  另外吗，对于有路由器的情况，接收端地址往往不是自己的地址，此时，需要借助路由控制表，在调查应该送往的主机或路由器之后再进行转发数据。
* ⑦ TCP 模块的处理  
  在 TCP 模块中，首先会计算一下校验和，判断数据是否被破坏。然后检查是否在按照序号接收数据。最后检查端口号，确定具体的应用程序。数据被完整地接收以后，会传给由端口号识别的应用程序。
* ⑧ 应用程序的处理  
  接收端应用程序会直接接收发送端发送的数据。通过解析数据，展示相应的内容。

**三、传输层中的 TCP 和 UDP**

TCP/IP 中有两个具有代表性的传输层协议，分别是 TCP 和 UDP。

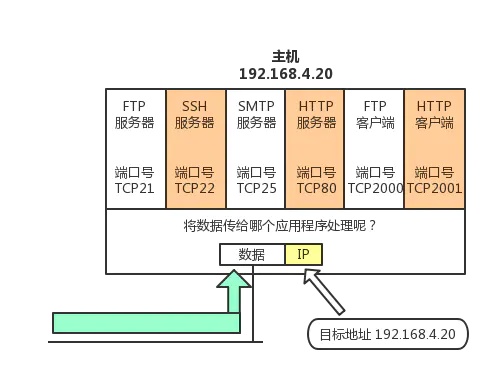
* TCP 是面向连接的、可靠的流协议。流就是指不间断的数据结构，当应用程序采用 TCP 发送消息时，虽然可以保证发送的顺序，但还是犹如没有任何间隔的数据流发送给接收端。TCP 为提供可靠性传输，实行“顺序控制”或“重发控制”机制。此外还具备“流控制（流量控制）”、“拥塞控制”、提高网络利用率等众多功能。
* UDP 是不具有可靠性的数据报协议。细微的处理它会交给上层的应用去完成。在 UDP 的情况下，虽然可以确保发送消息的大小，却不能保证消息一定会到达。因此，应用有时会根据自己的需要进行重发处理。
* TCP 和 UDP 的优缺点无法简单地、绝对地去做比较：TCP 用于在传输层有必要实现可靠传输的情况；而在一方面，UDP 主要用于那些对高速传输和实时性有较高要求的通信或广播通信。TCP 和 UDP 应该根据应用的目的按需使用。

**1. 端口号**

数据链路和 IP 中的地址，分别指的是 MAC 地址和 IP 地址。前者用来识别同一链路中不同的计算机，后者用来识别 TCP/IP 网络中互连的主机和路由器。在传输层也有这种类似于地址的概念，那就是端口号。端口号用来识别同一台计算机中进行通信的不同应用程序。因此，它也被称为程序地址。

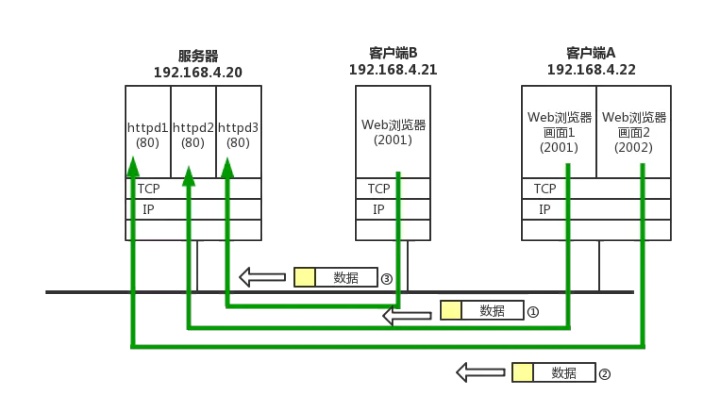
**1.1 根据端口号识别应用**

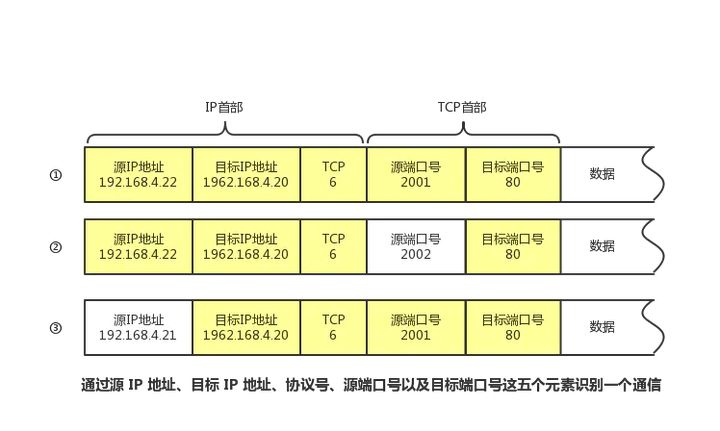
一台计算机上同时可以运行多个程序。传输层协议正是利用这些端口号识别本机中正在进行通信的应用程序，并准确地将数据传输。



**1.2 通过 IP 地址、端口号、协议号进行通信识别**

* 仅凭目标端口号识别某一个通信是远远不够的。





* ① 和② 的通信是在两台计算机上进行的。它们的目标端口号相同，都是80。这里可以根据源端口号加以区分。
* ③ 和 ① 的目标端口号和源端口号完全相同，但它们各自的源 IP 地址不同。
* 此外，当 IP 地址和端口号全都一样时，我们还可以通过协议号来区分（TCP 和 UDP）。

**1.3 端口号的确定**

* 标准既定的端口号：这种方法也叫静态方法。它是指每个应用程序都有其指定的端口号。但并不是说可以随意使用任何一个端口号。例如 HTTP、FTP、TELNET 等广为使用的应用协议中所使用的端口号就是固定的。这些端口号被称为知名端口号，分布在 0~1023 之间；除知名端口号之外，还有一些端口号被正式注册，它们分布在 1024~49151 之间，不过这些端口号可用于任何通信用途。
* 时序分配法：服务器有必要确定监听端口号，但是接受服务的客户端没必要确定端口号。在这种方法下，客户端应用程序完全可以不用自己设置端口号，而全权交给操作系统进行分配。动态分配的端口号范围在 49152~65535 之间。

**1.4 端口号与协议**

* 端口号由其使用的传输层协议决定。因此，不同的传输层协议可以使用相同的端口号。
* 此外，那些知名端口号与传输层协议并无关系。只要端口一致都将分配同一种应用程序进行处理。

**2. UDP**

* UDP 不提供复杂的控制机制，利用 IP 提供面向无连接的通信服务。
* 并且它是将应用程序发来的数据在收到的那一刻，立即按照原样发送到网络上的一种机制。即使是出现网络拥堵的情况，UDP 也无法进行流量控制等避免网络拥塞行为。
* 此外，传输途中出现丢包，UDP 也不负责重发。
* 甚至当包的到达顺序出现乱序时也没有纠正的功能。
* 如果需要以上的细节控制，不得不交由采用 UDP 的应用程序去处理。
* UDP 常用于一下几个方面：1.包总量较少的通信（DNS、SNMP等）；2.视频、音频等多媒体通信（即时通信）；3.限定于 LAN 等特定网络中的应用通信；4.广播通信（广播、多播）。

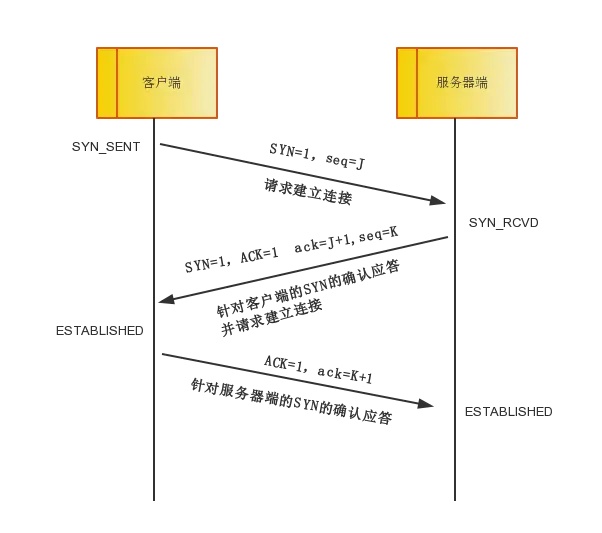
**3. TCP**

* TCP 与 UDP 的区别相当大。它充分地实现了数据传输时各种控制功能，可以进行丢包时的重发控制，还可以对次序乱掉的分包进行顺序控制。而这些在 UDP 中都没有。
* 此外，TCP 作为一种面向有连接的协议，只有在确认通信对端存在时才会发送数据，从而可以控制通信流量的浪费。
* 根据 TCP 的这些机制，在 IP 这种无连接的网络上也能够实现高可靠性的通信（ 主要通过检验和、序列号、确认应答、重发控制、连接管理以及窗口控制等机制实现）。

**3.1 三次握手（重点）**

* TCP 提供面向有连接的通信传输。面向有连接是指在数据通信开始之前先做好两端之间的准备工作。
* 所谓三次握手是指建立一个 TCP 连接时需要客户端和服务器端总共发送三个包以确认连接的建立。在socket编程中，这一过程由客户端执行connect来触发。

下面来看看三次握手的流程图：

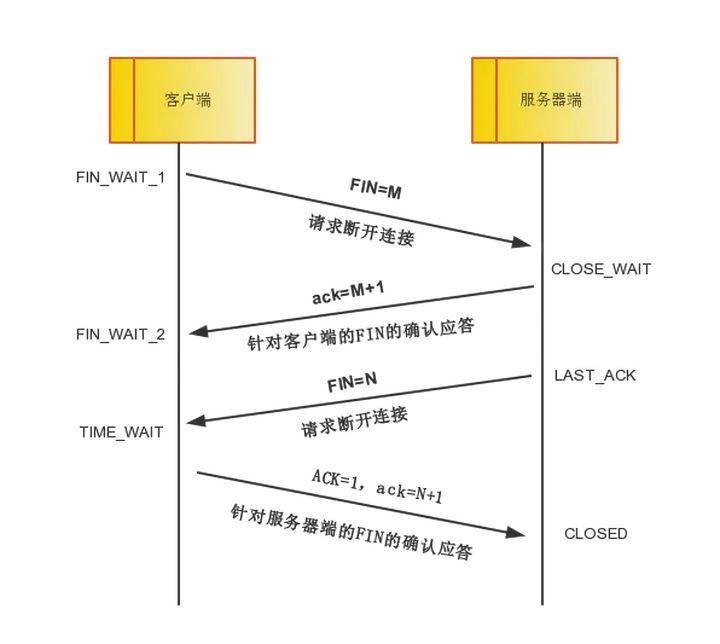


* 第一次握手：客户端将标志位SYN置为1，随机产生一个值seq=J，并将该数据包发送给服务器端，客户端进入SYN\_SENT状态，等待服务器端确认。
* 第二次握手：服务器端收到数据包后由标志位SYN=1知道客户端请求建立连接，服务器端将标志位SYN和ACK都置为1，ack=J+1，随机产生一个值seq=K，并将该数据包发送给客户端以确认连接请求，服务器端进入SYN\_RCVD状态。
* 第三次握手：客户端收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给服务器端，服务器端检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，客户端和服务器端进入ESTABLISHED状态，完成三次握手，随后客户端与服务器端之间可以开始传输数据了。

**3.2 四次挥手（重点）**

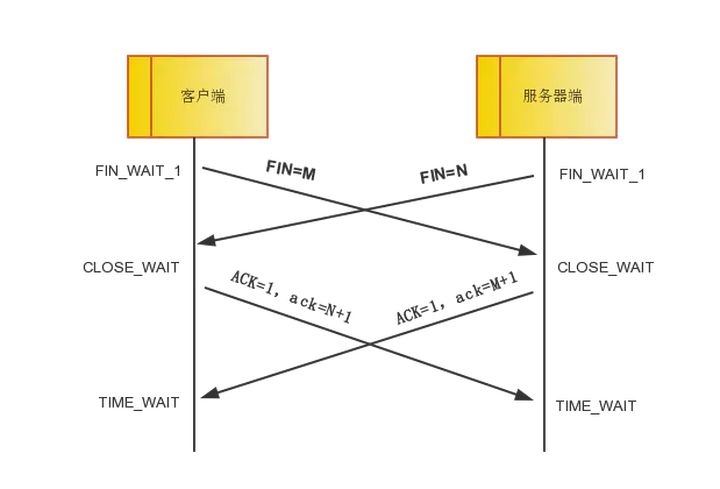
* 四次挥手即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认连接的断开。在socket编程中，这一过程由客户端或服务端任一方执行close来触发。
* 由于TCP连接是全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭。

下面来看看四次挥手的流程图：



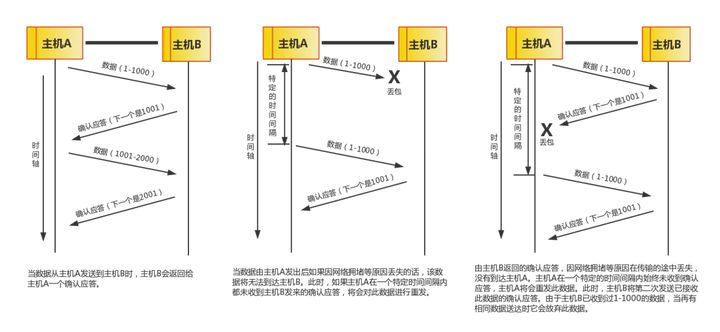
* 中断连接端可以是客户端，也可以是服务器端。
* 第一次挥手：客户端发送一个FIN=M，用来关闭客户端到服务器端的数据传送，客户端进入FIN\_WAIT\_1状态。意思是说"我客户端没有数据要发给你了"，但是如果你服务器端还有数据没有发送完成，则不必急着关闭连接，可以继续发送数据。
* 第二次挥手：服务器端收到FIN后，先发送ack=M+1，告诉客户端，你的请求我收到了，但是我还没准备好，请继续你等我的消息。这个时候客户端就进入FIN\_WAIT\_2 状态，继续等待服务器端的FIN报文。
* 第三次挥手：当服务器端确定数据已发送完成，则向客户端发送FIN=N报文，告诉客户端，好了，我这边数据发完了，准备好关闭连接了。服务器端进入LAST\_ACK状态。
* 第四次挥手：客户端收到FIN=N报文后，就知道可以关闭连接了，但是他还是不相信网络，怕服务器端不知道要关闭，所以发送ack=N+1后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。服务器端收到ACK后，就知道可以断开连接了。客户端等待了2MSL后依然没有收到回复，则证明服务器端已正常关闭，那好，我客户端也可以关闭连接了。最终完成了四次握手。

**上面是一方主动关闭，另一方被动关闭的情况，实际中还会出现同时发起主动关闭的情况，**  
具体流程如下图：



**3.3 通过序列号与确认应答提高可靠性**

* 在 TCP 中，当发送端的数据到达接收主机时，接收端主机会返回一个已收到消息的通知。这个消息叫做确认应答（ACK）。当发送端将数据发出之后会等待对端的确认应答。如果有确认应答，说明数据已经成功到达对端。**反之，则数据丢失的可能性很大**。
* 在一定时间内没有等待到确认应答，发送端就可以认为数据已经丢失，并进行重发。由此，即使产生了丢包，仍然能够保证数据能够到达对端，实现可靠传输。
* 未收到确认应答并不意味着数据一定丢失。也有可能是数据对方已经收到，只是返回的确认应答在途中丢失。这种情况也会导致发送端误以为数据没有到达目的地而重发数据。
* 此外，也有可能因为一些其他原因导致确认应答延迟到达，在源主机重发数据以后才到达的情况也屡见不鲜。此时，源主机只要按照机制重发数据即可。
* 对于目标主机来说，反复收到相同的数据是不可取的。为了对上层应用提供可靠的传输，目标主机必须放弃重复的数据包。为此我们引入了序列号。
* **序列号是按照顺序给发送数据的每一个字节（8位字节）都标上号码的编号。接收端查询接收数据 TCP 首部中的序列号和数据的长度，将自己下一步应该接收的序列号作为确认应答返送回去。通过序列号和确认应答号，TCP 能够识别是否已经接收数据，又能够判断是否需要接收，从而实现可靠传输。**



**3.4 重发超时的确定**

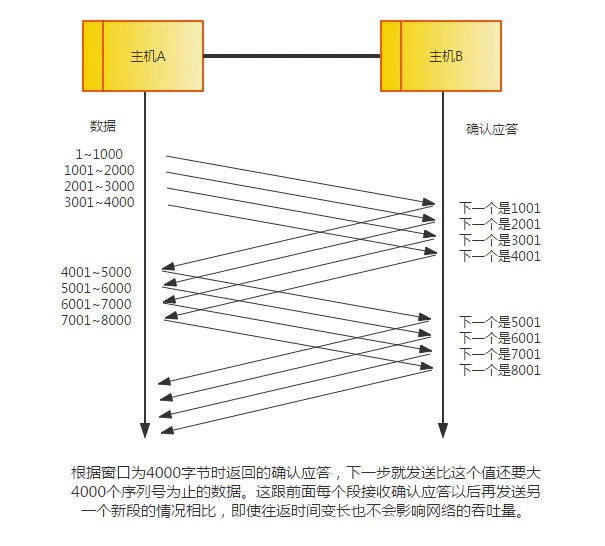
* **重发超时是指在重发数据之前，等待确认应答到来的那个特定时间间隔。**如果超过这个时间仍未收到确认应答，发送端将进行数据重发。最理想的是，找到一个最小时间，它能保证“确认应答一定能在这个时间内返回”。
* TCP 要求不论处在何种网络环境下都要提供高性能通信，并且无论网络拥堵情况发生何种变化，都必须保持这一特性。为此，它在每次发包时都会计算往返时间及其偏差。将这个往返时间和偏差时间相加，重发超时的时间就是比这个总和要稍大一点的值。
* 在 BSD 的 Unix 以及 Windows 系统中，超时都以0.5秒为单位进行控制，因此重发超时都是0.5秒的整数倍。不过，最初其重发超时的默认值一般设置为6秒左右。
* 数据被重发之后若还是收不到确认应答，则进行再次发送。此时，等待确认应答的时间将会以2倍、4倍的指数函数延长。
* 此外，**数据也不会被无限、反复地重发。达到一定重发次数之后，如果仍没有任何确认应答返回，就会判断为网络或对端主机发生了异常，强制关闭连接。并且通知应用通信异常强行终止。**

**3.5 以段为单位发送数据**

* 在建立 TCP 连接的同时，也可以确定发送数据包的单位，我们也可以称其为“最大消息长度”（MSS）。最理想的情况是，最大消息长度正好是 IP 中不会被分片处理的最大数据长度。
* TCP 在传送大量数据时，是以 MSS 的大小将数据进行分割发送。进行重发时也是以 MSS 为单位。
* MSS 在三次握手的时候，在两端主机之间被计算得出。两端的主机在发出建立连接的请求时，会在 TCP 首部中写入 MSS 选项，告诉对方自己的接口能够适应的 MSS 的大小。然后会在两者之间选择一个较小的值投入使用。

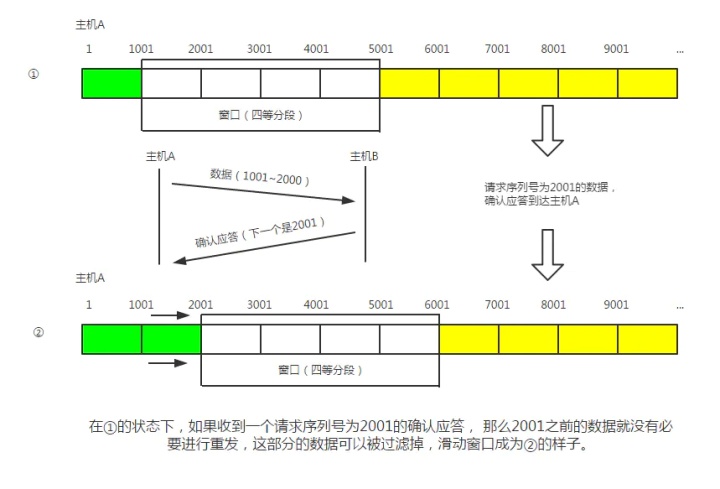
**3.6 利用窗口控制提高速度**

* TCP 以1个段为单位，每发送一个段进行一次确认应答的处理。这样的传输方式有一个缺点，就是包的往返时间越长通信性能就越低。
* 为解决这个问题，TCP 引入了窗口这个概念。确认应答不再是以每个分段，而是以更大的单位进行确认，转发时间将会被大幅地缩短。也就是说，发送端主机，在发送了一个段以后不必要一直等待确认应答，而是继续发送。如下图所示：



* 窗口大小就是指无需等待确认应答而可以继续发送数据的最大值。上图中窗口大小为4个段。这个机制实现了使用大量的缓冲区，通过对多个段同时进行确认应答的功能。

**3.7 滑动窗口控制**

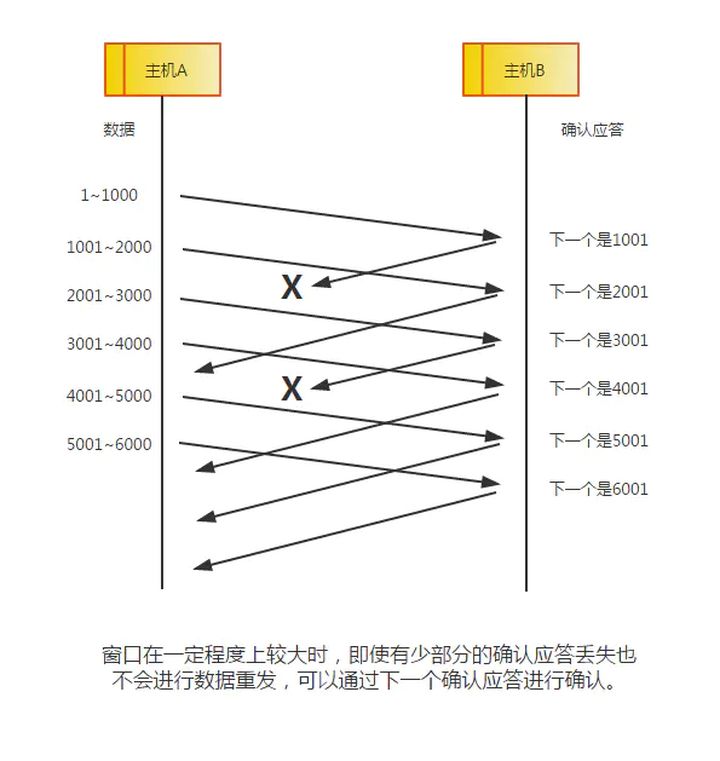


* 上图中的窗口内的数据即便没有收到确认应答也可以被发送出去。不过，在整个窗口的确认应答没有到达之前，如果其中部分数据出现丢包，那么发送端仍然要负责重传。为此，发送端主机需要设置缓存保留这些待被重传的数据，直到收到他们的确认应答。
* 在滑动窗口以外的部分包括未发送的数据以及已经确认对端已收到的数据。当数据发出后若如期收到确认应答就可以不用再进行重发，此时数据就可以从缓存区清除。
* 收到确认应答的情况下，将窗口滑动到确认应答中的序列号的位置。这样可以顺序地将多个段同时发送提高通信性能。这种机制也别称为滑动窗口控制。

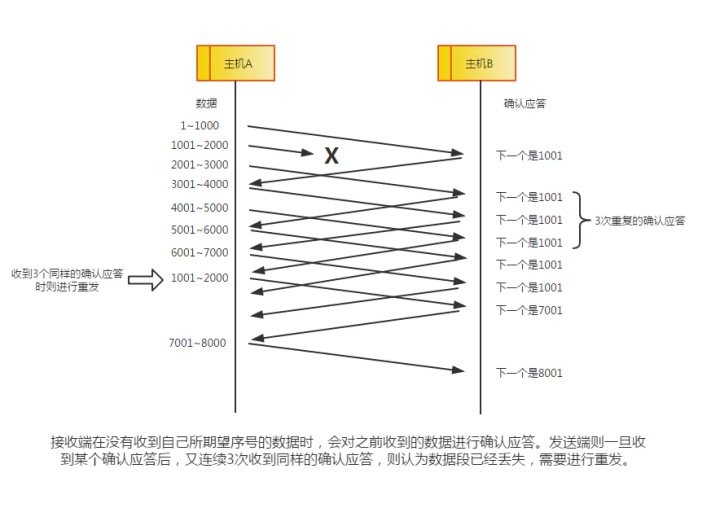
**3.8 窗口控制中的重发控制**

在使用窗口控制中， 出现丢包一般分为两种情况：

* ① 确认应答未能返回的情况。在这种情况下，数据已经到达对端，是不需要再进行重发的，如下图：



* ② 某个报文段丢失的情况。接收主机如果收到一个自己应该接收的序列号以外的数据时，会针对当前为止收到数据返回确认应答。如下图所示，当某一报文段丢失后，发送端会一直收到序号为1001的确认应答，因此，在窗口比较大，又出现报文段丢失的情况下，同一个序列号的确认应答将会被重复不断地返回。而发送端主机如果连续3次收到同一个确认应答，就会将其对应的数据进行重发。这种机制比之前提到的超时管理更加高效，因此也被称为高速重发控制。



**四、网络层中的 IP 协议**

* IP（IPv4、IPv6）相当于 OSI 参考模型中的第3层——网络层。网络层的主要作用是“实现终端节点之间的通信”。这种终端节点之间的通信也叫“点对点通信”。
* 网络的下一层——数据链路层的主要作用是在互连同一种数据链路的节点之间进行包传递。而一旦跨越多种数据链路，就需要借助网络层。网络层可以跨越不同的数据链路，即使是在不同的数据链路上也能实现两端节点之间的数据包传输。
* **IP 大致分为三大作用模块，它们是 IP 寻址、路由（最终节点为止的转发）以及 IP 分包与组包。**

**1. IP 地址**

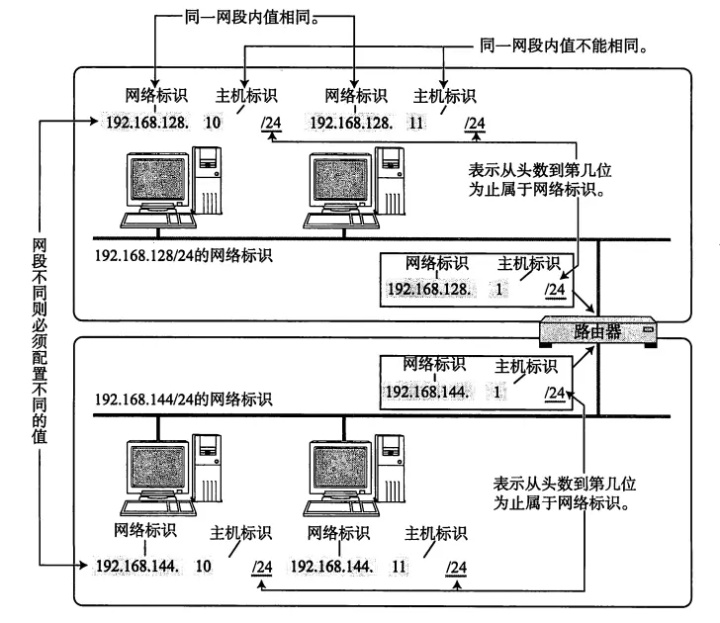
**1.1 IP 地址概述**

* 在计算机通信中，为了识别通信对端，必须要有一个类似于地址的识别码进行标识。在数据链路中的 MAC 地址正是用来标识同一个链路中不同计算机的一种识别码。
* 作为网络层的 IP ,也有这种地址信息，一般叫做 IP 地址。IP 地址用于在“连接到网络中的所有主机中识别出进行通信的目标地址”。因此，在 TCP/IP 通信中所有主机或路由器必须设定自己的 IP 地址。
* 不论一台主机与哪种数据链路连接，其 IP 地址的形式都保持不变。
* IP 地址（IPv4 地址）由32位正整数来表示。IP 地址在计算机内部以二进制方式被处理。然而，由于我们并不习惯于采用二进制方式，我们将32位的 IP 地址以每8位为一组，分成4组，每组以 “.” 隔开，再将每组数转换成十进制数。如下：

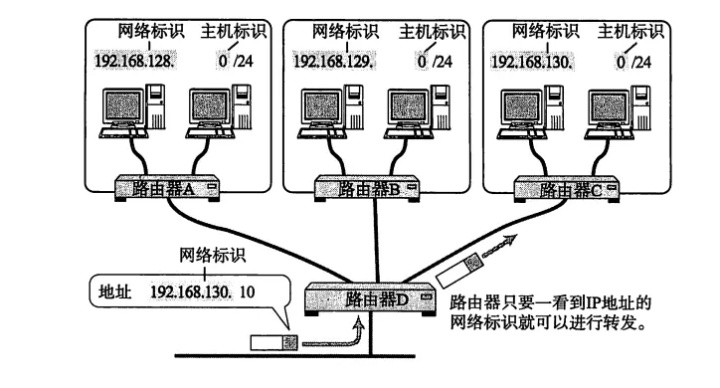
2828282810101100000101000000000100000001（2进制）10101100.00010100.00000001.00000001（2进制）172.20.1.1（10进制）

**1.2 IP 地址由网络和主机两部分标识组成**

* 如下图，网络标识在数据链路的每个段配置不同的值。网络标识必须保证相互连接的每个段的地址不相重复。而相同段内相连的主机必须有相同的网络地址。IP 地址的“主机标识”则不允许在同一个网段内重复出现。由此，可以通过设置网络地址和主机地址，在相互连接的整个网络中保证每台主机的 IP 地址都不会相互重叠。即 IP 地址具有了唯一性。



* 如下图，IP 包被转发到途中某个路由器时，正是利用目标 IP 地址的网络标识进行路由。因为即使不看主机标识，只要一见到网络标识就能判断出是否为该网段内的主机。



**1.3 IP 地址的分类**

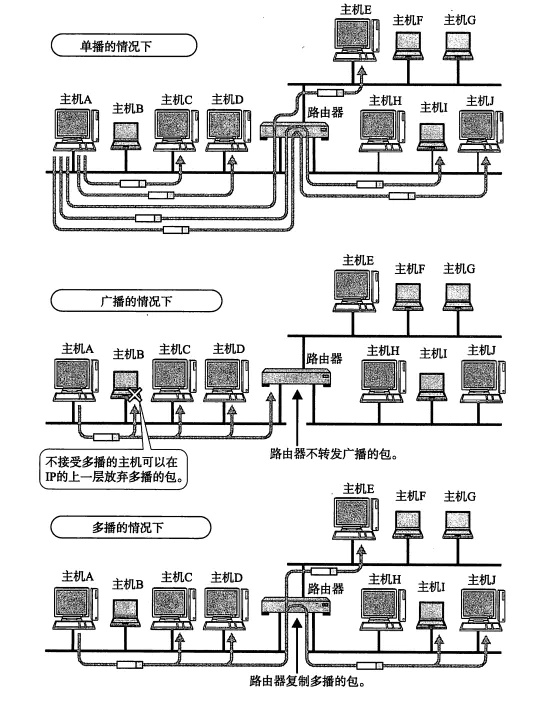
* **IP 地址分为四个级别，分别为A类、B类、C类、D类。它根据 IP 地址中从第 1 位到第 4 位的比特列对其网络标识和主机标识进行区分。**
* **A 类 IP 地址是首位以 “0” 开头的地址。**从第 1 位到第 8 位是它的网络标识。用十进制表示的话，0.0.0.0~127.0.0.0 是 A 类的网络地址。A 类地址的后 24 位相当于主机标识。因此，一个网段内可容纳的主机地址上限为16,777,214个。
* **B 类 IP 地址是前两位 “10” 的地址。**从第 1 位到第 16 位是它的网络标识。用十进制表示的话，128.0.0.0~191.255.0.0 是 B 类的网络地址。B 类地址的后 16 位相当于主机标识。因此，一个网段内可容纳的主机地址上限为65,534个。
* **C 类 IP 地址是前三位为 “110” 的地址。**从第 1 位到第 24 位是它的网络标识。用十进制表示的话，192.0.0.0~223.255.255.0 是 C 类的网络地址。C 类地址的后 8 位相当于主机标识。因此，一个网段内可容纳的主机地址上限为254个。
* **D 类 IP 地址是前四位为 “1110” 的地址。**从第 1 位到第 32 位是它的网络标识。用十进制表示的话，224.0.0.0~239.255.255.255 是 D 类的网络地址。D 类地址没有主机标识，常用于多播。
* 在分配 IP 地址时关于主机标识有一点需要注意。即要用比特位表示主机地址时，不可以全部为 0 或全部为 1。因为全部为 0 只有在表示对应的网络地址或 IP 地址不可以获知的情况下才使用。而全部为 1 的主机通常作为广播地址。因此，在分配过程中，应该去掉这两种情况。这也是为什么 C 类地址每个网段最多只能有 254（ 28 - 2 = 254）个主机地址的原因。

**1.4 广播地址**

* 广播地址用于在同一个链路中相互连接的主机之间发送数据包。将 IP 地址中的主机地址部分全部设置为 1，就成了广播地址。
* 广播分为本地广播和直接广播两种。在本网络内的广播叫做本地广播；在不同网络之间的广播叫做直接广播。

**1.5 IP 多播**

* 多播用于将包发送给特定组内的所有主机。由于其直接使用 IP 地址，因此也不存在可靠传输。
* 相比于广播，多播既可以穿透路由器，又可以实现只给那些必要的组发送数据包。请看下图：



* 多播使用 D 类地址。因此，如果从首位开始到第 4 位是 “1110”，就可以认为是多播地址。而剩下的 28 位可以成为多播的组编号。
* 此外， 对于多播，所有的主机（路由器以外的主机和终端主机）必须属于 224.0.0.1 的组，所有的路由器必须属于 224.0.0.2 的组。

**1.6 子网掩码**

* 现在一个 IP 地址的网络标识和主机标识已不再受限于该地址的类别，而是由一个叫做“子网掩码”的识别码通过子网网络地址细分出比 A 类、B 类、C 类更小粒度的网络。这种方式实际上就是将原来 A 类、B 类、C 类等分类中的主机地址部分用作子网地址，可以将原网络分为多个物理网络的一种机制。
* 子网掩码用二进制方式表示的话，也是一个 32 位的数字。它对应 IP 地址网络标识部分的位全部为 “1”，对应 IP 地址主机标识的部分则全部为 “0”。由此，一个 IP 地址可以不再受限于自己的类别，而是可以用这样的子网掩码自由地定位自己的网络标识长度。当然，子网掩码必须是 IP 地址的首位开始连续的 “1”。
* 对于子网掩码，目前有两种表示方式。第一种是，将 IP 地址与子网掩码的地址分别用两行来表示。以 172.20.100.52 的前 26 位是网络地址的情况为例，如下：

IP 地址172.20.100.52子网掩码255.255.255.192网络地址172.20.100.0子网掩码255.255.255.192广播地址172.20.100.63子网掩码255.255.255.192

* 第二种表示方式是，在每个 IP 地址后面追加网络地址的位数用 “/ ” 隔开，如下：

IP 地址172.20.100.52/ 26网络地址172.20.100.0/ 26广播地址172.20.100.63/ 26

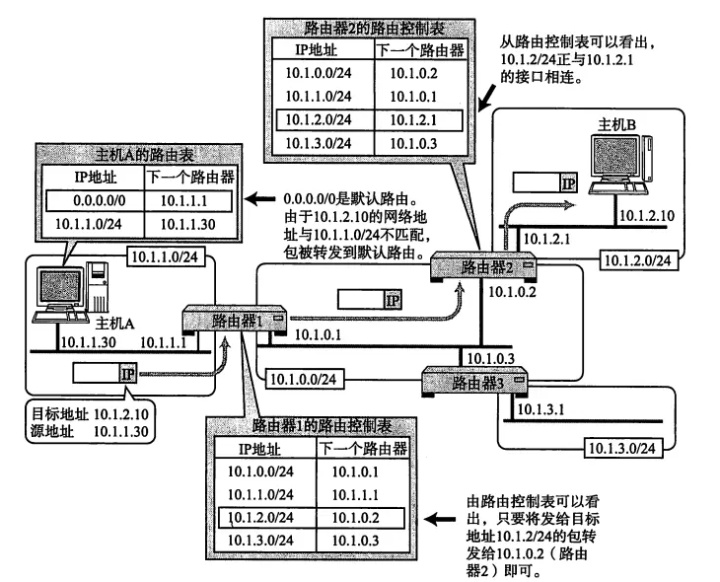
* 另外，在第二种方式下记述网络地址时可以省略后面的 “0” 。例如：172.20.0.0/26 跟 172.20/26 其实是一个意思。

**2. 路由**

* 发送数据包时所使用的地址是网络层的地址，即 IP 地址。然而仅仅有 IP 地址还不足以实现将数据包发送到对端目标地址，在数据发送过程中还需要类似于“指明路由器或主机”的信息，以便真正发往目标地址。保存这种信息的就是路由控制表。
* 该路由控制表的形成方式有两种：一种是管理员手动设置，另一种是路由器与其他路由器相互交换信息时自动刷新。前者也叫做静态路由控制，而后者叫做动态路由控制。
* IP 协议始终认为路由表是正确的。然后，IP 本身并没有定义制作路由控制表的协议。即 IP 没有制作路由控制表的机制。该表示由一个叫做“路由协议”的协议制作而成。

**2.1 IP 地址与路由控制**

* IP 地址的网络地址部分用于进行路由控制。
* 路由控制表中记录着网络地址与下一步应该发送至路由器的地址。
* 在发送 IP 包时，首先要确定 IP 包首部中的目标地址，再从路由控制表中找到与该地址具有相同网络地址的记录，根据该记录将 IP 包转发给相应的下一个路由器。如果路由控制表中存在多条相同网络地址的记录，就选择一个最为吻合的网络地址。



**3. IP 分包与组包**

* 每种数据链路的最大传输单元（MTU）都不尽相同，因为每个不同类型的数据链路的使用目的不同。使用目的不同，可承载的 MTU 也就不同。
* 任何一台主机都有必要对 IP 分片进行相应的处理。分片往往在网络上遇到比较大的报文无法一下子发送出去时才会进行处理。
* 经过分片之后的 IP 数据报在被重组的时候，只能由目标主机进行。路由器虽然做分片但不会进行重组。

**3.1 路径 MTU 发现**

* 分片机制也有它的不足。如路由器的处理负荷加重之类。因此，只要允许，是不希望由路由器进行 IP 数据包的分片处理的。
* 为了应对分片机制的不足，“路径 MTU 发现” 技术应运而生。路径 MTU 指的是，从发送端主机到接收端主机之间不需要分片是最大 MTU 的大小。即路径中存在的所有数据链路中最小的 MTU 。
* 进行路径 MTU 发现，就可以避免在中途的路由器上进行分片处理，也可以在 TCP 中发送更大的包。

**4. IPv6**

* IPv6（IP version 6）是为了根本解决 IPv4 地址耗尽的问题而被标准化的网际协议。IPv4 的地址长度为 4 个 8 位字节，即 32 比特。而 IPv6 的地址长度则是原来的 4 倍，即 128 比特，一般写成 8 个 16 位字节。

**4.1 IPv6 的特点**

* IP 得知的扩大与路由控制表的聚合。
* 性能提升。包首部长度采用固定的值（40字节），不再采用首部检验码。简化首部结构，减轻路由器负担。路由器不再做分片处理。
* 支持即插即用功能。即使没有DHCP服务器也可以实现自动分配 IP 地址。
* 采用认证与加密功能。应对伪造 IP 地址的网络安全功能以及防止线路窃听的功能。
* 多播、Mobile IP 成为扩展功能。

**4.2 IPv6 中 IP 地址的标记方法**

* 一般人们将 128 比特 IP 地址以每 16 比特为一组，每组用冒号（“：”）隔开进行标记。
* 而且如果出现连续的 0 时还可以将这些 0 省略，并用两个冒号（“：：”）隔开。但是，一个 IP 地址中只允许出现一次两个连续的冒号。

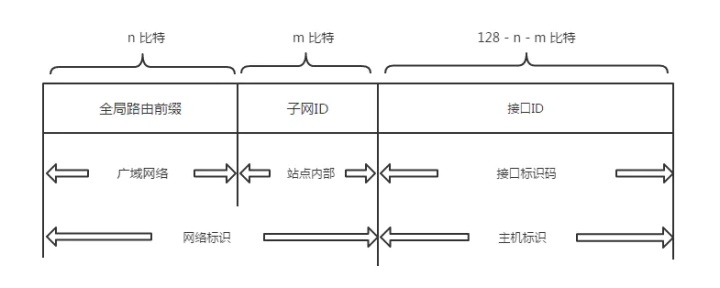
**4.3 IPv6 地址的结构**

* IPv6 类似 IPv4，也是通过 IP 地址的前几位标识 IP 地址的种类。
* 在互联网通信中，使用一种全局的单播地址。它是互联网中唯一的一个地址，不需要正式分配 IP 地址。

未定义0000 ... 0000（128比特）：：/ 128环回地址0000 ... 0001（128比特）：：1 / 128唯一本地地址1111 110FC00：/ 7链路本地单播地址1111 1110 10FE80：：/ 10多播地址1111 1111FF00：：/ 8全局单播地址（其他）

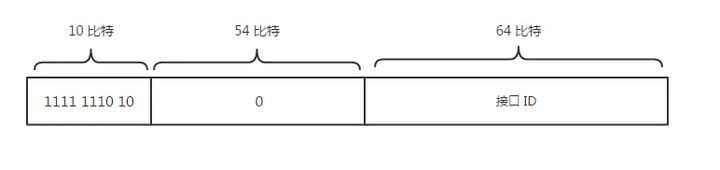
**4.4 全局单播地址**

* 全局单播地址是指世界上唯一的一个地址。它是互联网通信以及各个域内部通信中最为常用的一个 IPv6 地址。
* 格式如下图所示，现在 IPv6 的网络中所使用的格式为，n = 48，m = 16 以及 128 - n - m = 64。即前 64 比特为网络标识，后 64 比特为主机标识。



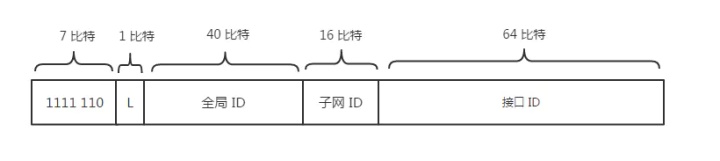
**4.5 链路本地单播地址**

* 链路本地单播地址是指在同一个数据链路内唯一的地址。它用于不经过路由器，在同一个链路中的通信。通常接口 ID 保存 64 比特版的 MAC 地址。



**4.6 唯一本地地址**

* 唯一本地地址是不进行互联网通信时所用的地址。
* 唯一本地地址虽然不会与互联网连接，但是也会尽可能地随机生成一个唯一的全局 ID。
* L 通常被置为 1
* 全局 ID 的值随机决定
* 子网 ID 是指该域子网地址
* 接口 ID 即为接口的 ID



**4.7 IPv6 分段处理**

* IPv6 的分片处理只在作为起点的发送端主机上进行，路由器不参与分片。
* IPv6 中最小 MTU 为 1280 字节，因此，在嵌入式系统中对于那些有一定系统资源限制的设备来说，不需要进行“路径 MTU 发现”，而是在发送 IP 包时直接以 1280 字节为单位分片送出。

**4.8 IP 首部（暂略）**

**5. IP 协议相关技术**

* IP 旨在让最终目标主机收到数据包，但是在这一过程中仅仅有 IP 是无法实现通信的。必须还有能够解析主机名称和 MAC 地址的功能，以及数据包在发送过程中异常情况处理的功能。

**5.1 DNS**

* 我们平常在访问某个网站时不适用 IP 地址，而是用一串由罗马字和点号组成的字符串。而一般用户在使用 TCP/IP 进行通信时也不使用 IP 地址。能够这样做是因为有了 DNS （Domain Name System）功能的支持。DNS 可以将那串字符串自动转换为具体的 IP 地址。
* 这种 DNS 不仅适用于 IPv4，还适用于 IPv6。

**5.2 ARP**

* 只要确定了 IP 地址，就可以向这个目标地址发送 IP 数据报。然而，在底层数据链路层，进行实际通信时却有必要了解每个 IP 地址所对应的 MAC 地址。
* ARP 是一种解决地址问题的协议。以目标 IP 地址为线索，用来定位下一个应该接收数据分包的网络设备对应的 MAC 地址。不过 ARP 只适用于 IPv4，不能用于 IPv6。IPv6 中可以用 ICMPv6 替代 ARP 发送邻居探索消息。
* RARP 是将 ARP 反过来，从 MAC 地址定位 IP 地址的一种协议。

**5.3 ICMP**

* ICMP 的主要功能包括，确认 IP 包是否成功送达目标地址，通知在发送过程当中 IP 包被废弃的具体原因，改善网络设置等。
* IPv4 中 ICMP 仅作为一个辅助作用支持 IPv4。也就是说，在 IPv4 时期，即使没有 ICMP，仍然可以实现 IP 通信。然而，在 IPv6 中，ICMP 的作用被扩大，如果没有 ICMPv6，IPv6 就无法进行正常通信。

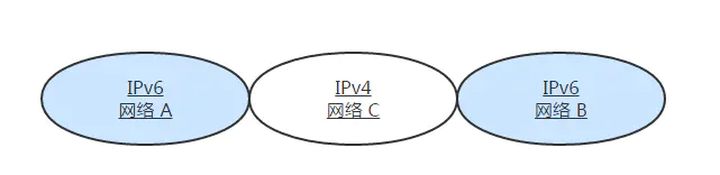
**5.4 DHCP**

* 如果逐一为每一台主机设置 IP 地址会是非常繁琐的事情。特别是在移动使用笔记本电脑、只能终端以及平板电脑等设备时，每移动到一个新的地方，都要重新设置 IP 地址。
* 于是，为了实现自动设置 IP 地址、统一管理 IP 地址分配，就产生了 DHCP（Dynamic Host Configuration Protocol）协议。有了 DHCP，计算机只要连接到网络，就可以进行 TCP/IP 通信。也就是说，DHCP 让即插即用变得可能。
* DHCP 不仅在 IPv4 中，在 IPv6 中也可以使用。

**5.5 NAT**

* NAT（Network Address Translator）是用于在本地网络中使用私有地址，在连接互联网时转而使用全局 IP 地址的技术。
* 除转换 IP 地址外，还出现了可以转换 TCP、UDP 端口号的 NAPT（Network Address Ports Translator）技术，由此可以实现用一个全局 IP 地址与多个主机的通信。
* NAT（NAPT）实际上是为正在面临地址枯竭的 IPv4 而开发的技术。不过，在 IPv6 中为了提高网络安全也在使用 NAT，在 IPv4 和 IPv6 之间的相互通信当中常常使用 NAT-PT。

**5.6 IP 隧道**



* 如上图的网络环境中，网络 A 与网络 B 之间无法直接进行通信，为了让它们之间正常通信，这时必须得采用 IP 隧道的功能。
* IP 隧道可以将那些从网络 A 发过来的 IPv6 的包统合为一个数据，再为之追加一个 IPv4 的首部以后转发给网络 C。
* 一般情况下，紧接着 IP 首部的是 TCP 或 UDP 的首部。然而，现在的应用当中“ IP 首部的后面还是 IP 首部”或者“ IP 首部的后面是 IPv6 的首部”等情况与日俱增。这种在网络层的首部后面追加网络层首部的通信方法就叫做“ IP 隧道”。