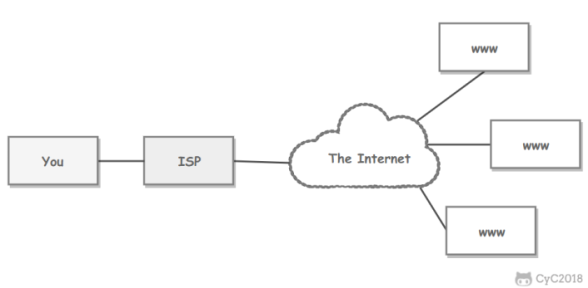
**概述**

**网络的网络**

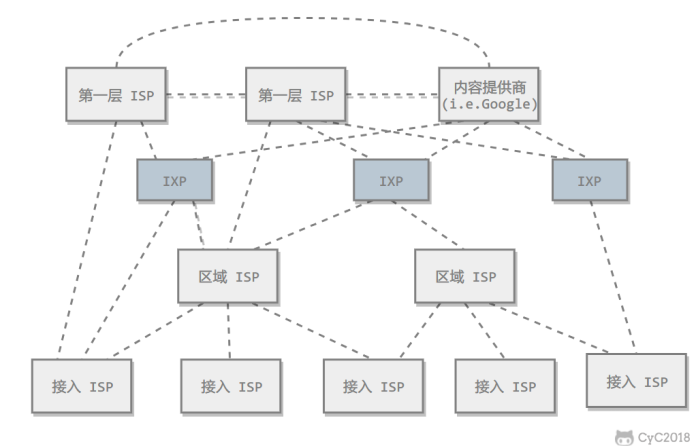
网络把主机连接起来，而互联网是把多种不同的网络连接起来，因此互联网是网络的网络。

**ISP**

互联网服务提供商 ISP 可以从互联网管理机构获得许多 IP 地址，同时拥有通信线路以及路由器等联网设备，个人或机构向 ISP 缴纳一定的费用就可以接入互联网。

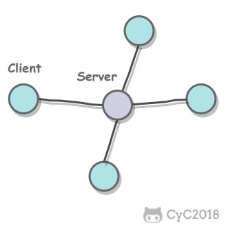
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/72be01cd-41ae-45f7-99b9-a8d284e44dd4.png)

目前的互联网是一种多层次 ISP 结构，ISP 根据覆盖面积的大小分为第一层 ISP、区域 ISP 和接入 ISP。互联网交换点 IXP 允许两个 ISP 直接相连而不用经过第三个 ISP。

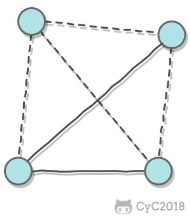
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/3be42601-9d33-4d29-8358-a9d16453af93.png)

**主机之间的通信方式**

* 客户-服务器（C/S）：客户是服务的请求方，服务器是服务的提供方。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/914894c2-0bc4-46b5-bef9-0316a69ef521.jpg)

* 对等（P2P）：不区分客户和服务器。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/42430e94-3137-48c0-bdb6-3cebaf9102e3.jpg)

**电路交换与分组交换**

**1. 电路交换**

电路交换用于电话通信系统，两个用户要通信之前需要建立一条专用的物理链路，并且在整个通信过程中始终占用该链路。由于通信的过程中不可能一直在使用传输线路，因此电路交换对线路的利用率很低，往往不到 10%。

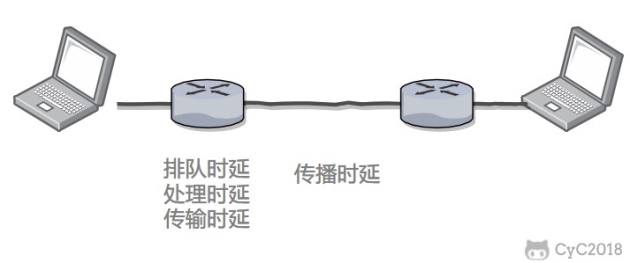
**2. 分组交换**

每个分组都有首部和尾部，包含了源地址和目的地址等控制信息，在同一个传输线路上同时传输多个分组互相不会影响，因此在同一条传输线路上允许同时传输多个分组，也就是说分组交换不需要占用传输线路。

在一个邮局通信系统中，邮局收到一份邮件之后，先存储下来，然后把相同目的地的邮件一起转发到下一个目的地，这个过程就是存储转发过程，分组交换也使用了存储转发过程。

**时延**

总时延 = 排队时延 + 处理时延 + 传输时延 + 传播时延

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/4b2ae78c-e254-44df-9e37-578e2f2bef52.jpg)

**1. 排队时延**

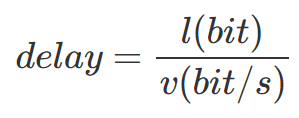
分组在路由器的输入队列和输出队列中排队等待的时间，取决于网络当前的通信量。

**2. 处理时延**

主机或路由器收到分组时进行处理所需要的时间，例如分析首部、从分组中提取数据、进行差错检验或查找适当的路由等。

**3. 传输时延**

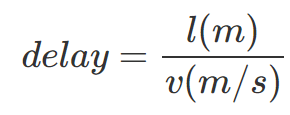
主机或路由器传输数据帧所需要的时间。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/dcdbb96c-9077-4121-aeb8-743e54ac02a4.png)

其中 l 表示数据帧的长度，v 表示传输速率。

**4. 传播时延**

电磁波在信道中传播所需要花费的时间，电磁波传播的速度接近光速。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/a1616dac-0e12-40b2-827d-9e3f7f0b940d.png)

其中 l 表示信道长度，v 表示电磁波在信道上的传播速度。

**计算机网络体系结构**

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/0fa6c237-a909-4e2a-a771-2c5485cd8ce0.png)

**1. 五层协议**

* **应用层** ：为特定应用程序提供数据传输服务，例如 HTTP、DNS 等协议。数据单位为报文。
* **传输层** ：为进程提供通用数据传输服务。由于应用层协议很多，定义通用的传输层协议就可以支持不断增多的应用层协议。运输层包括两种协议：传输控制协议 TCP，提供面向连接、可靠的数据传输服务，数据单位为报文段；用户数据报协议 UDP，提供无连接、尽最大努力的数据传输服务，数据单位为用户数据报。TCP 主要提供完整性服务，UDP 主要提供及时性服务。
* **网络层** ：为主机提供数据传输服务。而传输层协议是为主机中的进程提供数据传输服务。网络层把传输层传递下来的报文段或者用户数据报封装成分组。
* **数据链路层** ：网络层针对的还是主机之间的数据传输服务，而主机之间可以有很多链路，链路层协议就是为同一链路的主机提供数据传输服务。数据链路层把网络层传下来的分组封装成帧。
* **物理层** ：考虑的是怎样在传输媒体上传输数据比特流，而不是指具体的传输媒体。物理层的作用是尽可能屏蔽传输媒体和通信手段的差异，使数据链路层感觉不到这些差异。

**2. OSI**

其中表示层和会话层用途如下：

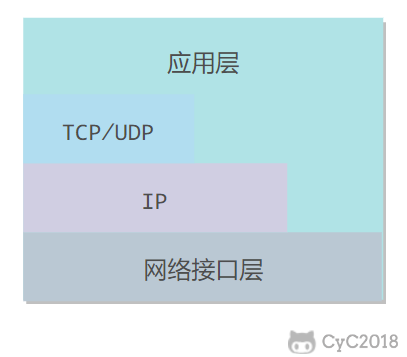
* **表示层** ：数据压缩、加密以及数据描述，这使得应用程序不必关心在各台主机中数据内部格式不同的问题。
* **会话层** ：建立及管理会话。

五层协议没有表示层和会话层，而是将这些功能留给应用程序开发者处理。

**3. TCP/IP**

它只有四层，相当于五层协议中数据链路层和物理层合并为网络接口层。

TCP/IP 体系结构不严格遵循 OSI 分层概念，应用层可能会直接使用 IP 层或者网络接口层。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/48d79be8-085b-4862-8a9d-18402eb93b31.png)

**4. 数据在各层之间的传递过程**

在向下的过程中，需要添加下层协议所需要的首部或者尾部，而在向上的过程中不断拆开首部和尾部。

路由器只有下面三层协议，因为路由器位于网络核心中，不需要为进程或者应用程序提供服务，因此也就不需要传输层和应用层。

物理层

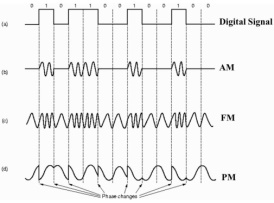
**通信方式**

根据信息在传输线上的传送方向，分为以下三种通信方式：

* 单工通信：单向传输
* 半双工通信：双向交替传输
* 全双工通信：双向同时传输

**带通调制**

模拟信号是连续的信号，数字信号是离散的信号。带通调制把数字信号转换为模拟信号。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/c34f4503-f62c-4043-9dc6-3e03288657df.jpg)

链路层

**基本问题**

**1. 封装成帧**

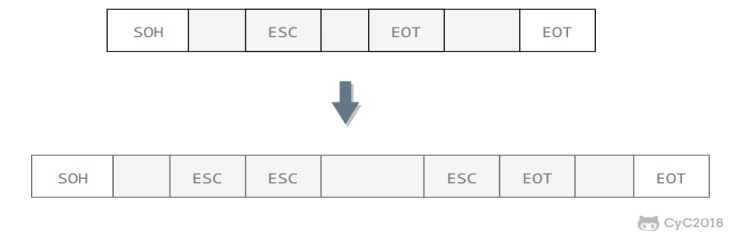
将网络层传下来的分组添加首部和尾部，用于标记帧的开始和结束。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/29a14735-e154-4f60-9a04-c9628e5d09f4.png)

**2. 透明传输**

透明表示一个实际存在的事物看起来好像不存在一样。

帧使用首部和尾部进行定界，如果帧的数据部分含有和首部尾部相同的内容，那么帧的开始和结束位置就会被错误的判定。需要在数据部分出现首部尾部相同的内容前面插入转义字符。如果数据部分出现转义字符，那么就在转义字符前面再加个转义字符。在接收端进行处理之后可以还原出原始数据。这个过程透明传输的内容是转义字符，用户察觉不到转义字符的存在。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/e738a3d2-f42e-4755-ae13-ca23497e7a97.png)

**3. 差错检测**

目前数据链路层广泛使用了循环冗余检验（CRC）来检查比特差错。

**信道分类**

**1. 广播信道**

一对多通信，一个节点发送的数据能够被广播信道上所有的节点接收到。

所有的节点都在同一个广播信道上发送数据，因此需要有专门的控制方法进行协调，避免发生冲突（冲突也叫碰撞）。

主要有两种控制方法进行协调，一个是使用信道复用技术，一是使用 CSMA/CD 协议。

**2. 点对点信道**

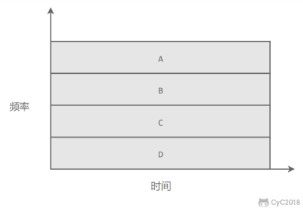
一对一通信。

因为不会发生碰撞，因此也比较简单，使用 PPP 协议进行控制。

**信道复用技术**

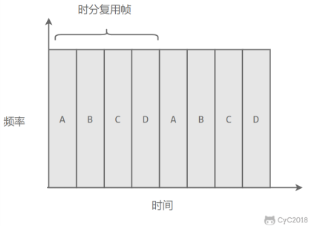
**1. 频分复用**

频分复用的所有主机在相同的时间占用不同的频率带宽资源。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/4aa5e057-bc57-4719-ab57-c6fbc861c505.png)

**2. 时分复用**

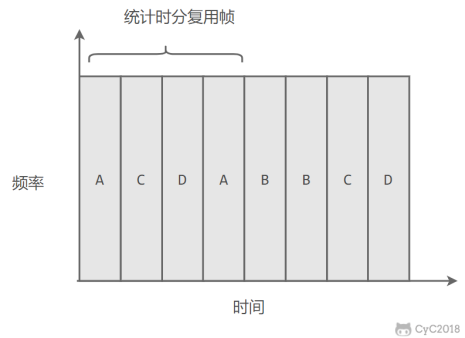
时分复用的所有主机在不同的时间占用相同的频率带宽资源。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/67582ade-d44a-46a6-8757-3c1296cc1ef9.png)

使用频分复用和时分复用进行通信，在通信的过程中主机会一直占用一部分信道资源。但是由于计算机数据的突发性质，通信过程没必要一直占用信道资源而不让出给其它用户使用，因此这两种方式对信道的利用率都不高。

**3. 统计时分复用**

是对时分复用的一种改进，不固定每个用户在时分复用帧中的位置，只要有数据就集中起来组成统计时分复用帧然后发送。

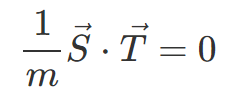
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/6283be2a-814a-4a10-84bf-9592533fe6bc.png)

**4. 波分复用**

光的频分复用。由于光的频率很高，因此习惯上用波长而不是频率来表示所使用的光载波。

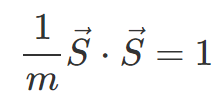
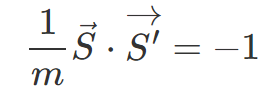
**5. 码分复用**

为每个用户分配 m bit 的码片，并且所有的码片正交，对于任意两个码片 [https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d](https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d) 和 [https://camo.githubusercontent.com/858d1274c608f6d519d53b7cb436dc8f08149571/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b547d](https://camo.githubusercontent.com/858d1274c608f6d519d53b7cb436dc8f08149571/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b547d) 有

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/308a02e9-3346-4251-8c41-bd5536dab491.png)

为了讨论方便，取 m=8，设码片 [https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d](https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d) 为 00011011。在拥有该码片的用户发送比特 1 时就发送该码片，发送比特 0 时就发送该码片的反码 11100100。

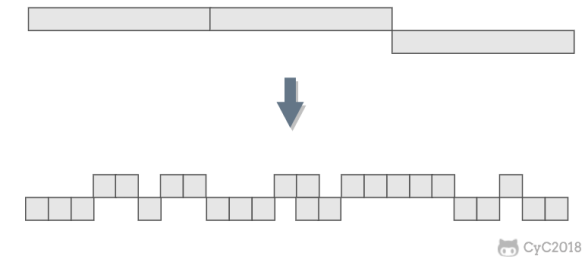
在计算时将 00011011 记作 (-1 -1 -1 +1 +1 -1 +1 +1)，可以得到

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/6fda1dc7-5c74-49c1-bb79-237a77e43a43.png)[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/e325a903-f0b1-4fbd-82bf-88913dc2f290.png)

其中 [https://camo.githubusercontent.com/a6ac16cdd1032e3879adb3216d357c68b506cf1d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b53277d](https://camo.githubusercontent.com/a6ac16cdd1032e3879adb3216d357c68b506cf1d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b53277d) 为 [https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d](https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d) 的反码。

利用上面的式子我们知道，当接收端使用码片 [https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d](https://camo.githubusercontent.com/375366f321c81ac7e67837c3fce1f1bc6f8a427d/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f5c7665637b537d) 对接收到的数据进行内积运算时，结果为 0 的是其它用户发送的数据，结果为 1 的是用户发送的比特 1，结果为 -1 的是用户发送的比特 0。

码分复用需要发送的数据量为原先的 m 倍。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/99b6060e-099d-4201-8e86-f8ab3768a7cf.png)

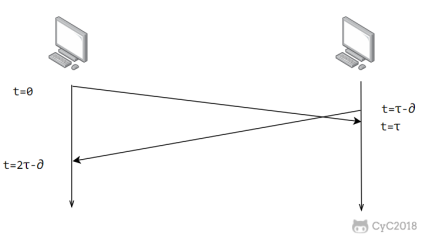
**CSMA/CD 协议**

CSMA/CD 表示载波监听多点接入 / 碰撞检测。

* **多点接入** ：说明这是总线型网络，许多主机以多点的方式连接到总线上。
* **载波监听** ：每个主机都必须不停地监听信道。在发送前，如果监听到信道正在使用，就必须等待。
* **碰撞检测** ：在发送中，如果监听到信道已有其它主机正在发送数据，就表示发生了碰撞。虽然每个主机在发送数据之前都已经监听到信道为空闲，但是由于电磁波的传播时延的存在，还是有可能会发生碰撞。

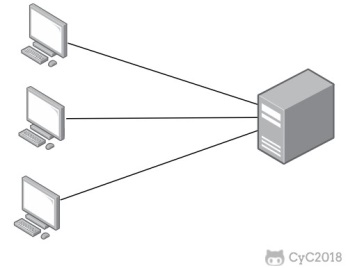
记端到端的传播时延为 τ，最先发送的站点最多经过 2τ 就可以知道是否发生了碰撞，称 2τ 为 **争用期** 。只有经过争用期之后还没有检测到碰撞，才能肯定这次发送不会发生碰撞。

当发生碰撞时，站点要停止发送，等待一段时间再发送。这个时间采用 **截断二进制指数退避算法** 来确定。从离散的整数集合 {0, 1, .., (2k-1)} 中随机取出一个数，记作 r，然后取 r 倍的争用期作为重传等待时间。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/19d423e9-74f7-4c2b-9b97-55890e0d5193.png)

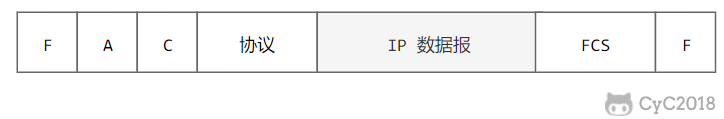
**PPP 协议**

互联网用户通常需要连接到某个 ISP 之后才能接入到互联网，PPP 协议是用户计算机和 ISP 进行通信时所使用的数据链路层协议。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/e1ab9f28-cb15-4178-84b2-98aad87f9bc8.jpg)

PPP 的帧格式：

* F 字段为帧的定界符
* A 和 C 字段暂时没有意义
* FCS 字段是使用 CRC 的检验序列
* 信息部分的长度不超过 1500

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/759013d7-61d8-4509-897a-d75af598a236.png)

**MAC 地址**

MAC 地址是链路层地址，长度为 6 字节（48 位），用于唯一标识网络适配器（网卡）。

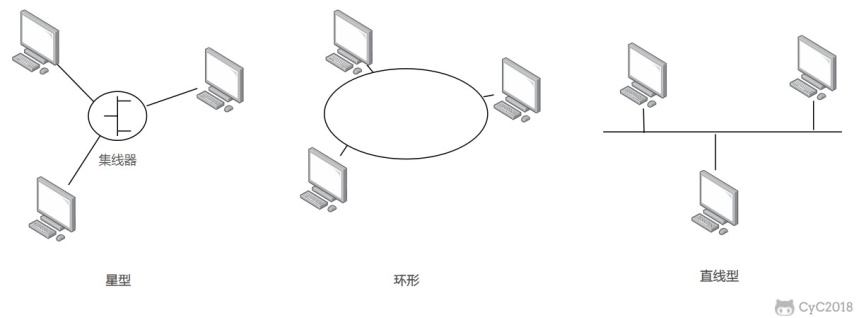
一台主机拥有多少个网络适配器就有多少个 MAC 地址。例如笔记本电脑普遍存在无线网络适配器和有线网络适配器，因此就有两个 MAC 地址。

**局域网**

局域网是一种典型的广播信道，主要特点是网络为一个单位所拥有，且地理范围和站点数目均有限。

主要有以太网、令牌环网、FDDI 和 ATM 等局域网技术，目前以太网占领着有线局域网市场。

可以按照网络拓扑结构对局域网进行分类：

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/807f4258-dba8-4c54-9c3c-a707c7ccffa2.jpg)

**以太网**

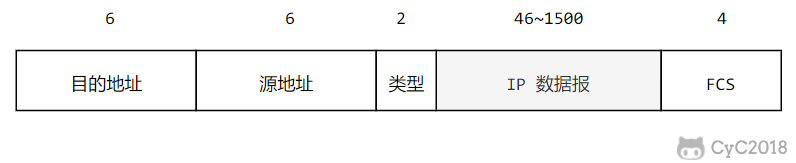
以太网是一种星型拓扑结构局域网。

早期使用集线器进行连接，集线器是一种物理层设备， 作用于比特而不是帧，当一个比特到达接口时，集线器重新生成这个比特，并将其能量强度放大，从而扩大网络的传输距离，之后再将这个比特发送到其它所有接口。如果集线器同时收到两个不同接口的帧，那么就发生了碰撞。

目前以太网使用交换机替代了集线器，交换机是一种链路层设备，它不会发生碰撞，能根据 MAC 地址进行存储转发。

以太网帧格式：

* **类型** ：标记上层使用的协议；
* **数据** ：长度在 46-1500 之间，如果太小则需要填充；
* **FCS** ：帧检验序列，使用的是 CRC 检验方法；

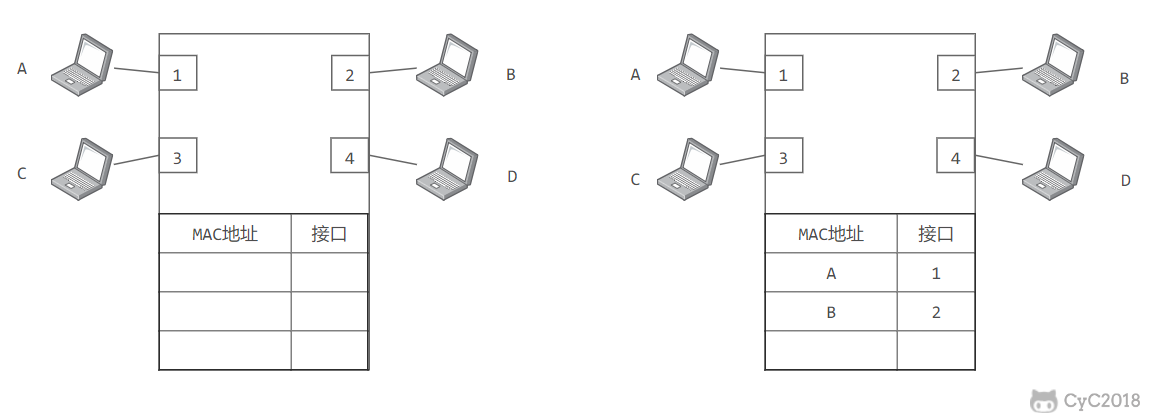
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/164944d3-bbd2-4bb2-924b-e62199c51b90.png)

**交换机**

交换机具有自学习能力，学习的是交换表的内容，交换表中存储着 MAC 地址到接口的映射。

正是由于这种自学习能力，因此交换机是一种即插即用设备，不需要网络管理员手动配置交换表内容。

下图中，交换机有 4 个接口，主机 A 向主机 B 发送数据帧时，交换机把主机 A 到接口 1 的映射写入交换表中。为了发送数据帧到 B，先查交换表，此时没有主机 B 的表项，那么主机 A 就发送广播帧，主机 C 和主机 D 会丢弃该帧，主机 B 回应该帧向主机 A 发送数据包时，交换机查找交换表得到主机 A 映射的接口为 1，就发送数据帧到接口 1，同时交换机添加主机 B 到接口 2 的映射。

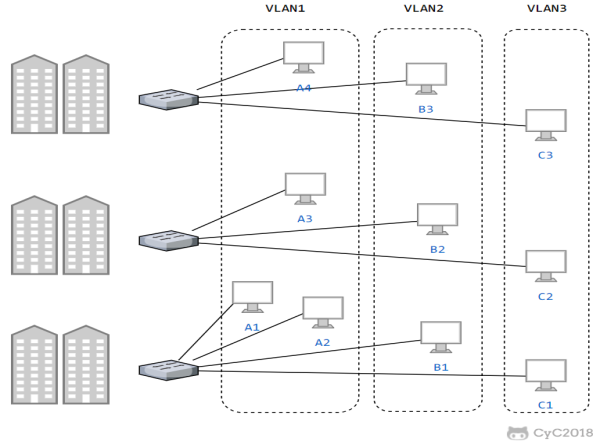
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/a4444545-0d68-4015-9a3d-19209dc436b3.png)

**虚拟局域网**

虚拟局域网可以建立与物理位置无关的逻辑组，只有在同一个虚拟局域网中的成员才会收到链路层广播信息。

例如下图中 (A1, A2, A3, A4) 属于一个虚拟局域网，A1 发送的广播会被 A2、A3、A4 收到，而其它站点收不到。

使用 VLAN 干线连接来建立虚拟局域网，每台交换机上的一个特殊接口被设置为干线接口，以互连 VLAN 交换机。IEEE 定义了一种扩展的以太网帧格式 802.1Q，它在标准以太网帧上加进了 4 字节首部 VLAN 标签，用于表示该帧属于哪一个虚拟局域网。

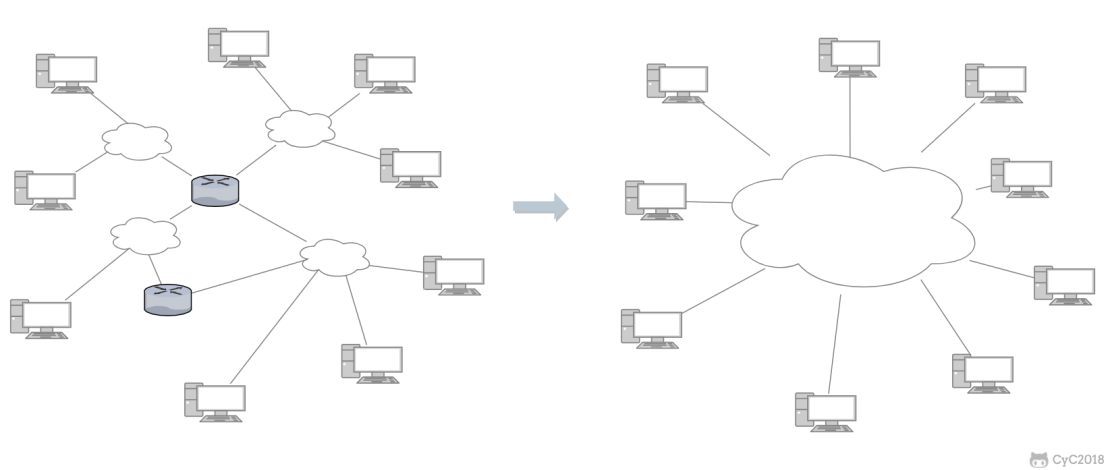
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/e98e9d20-206b-4533-bacf-3448d0096f38.png)

网络层

**概述**

因为网络层是整个互联网的核心，因此应当让网络层尽可能简单。网络层向上只提供简单灵活的、无连接的、尽最大努力交互的数据报服务。

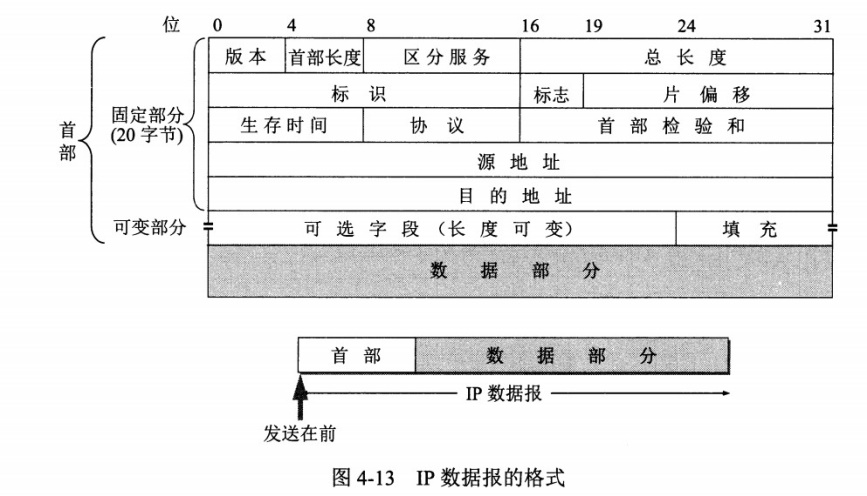
使用 IP 协议，可以把异构的物理网络连接起来，使得在网络层看起来好像是一个统一的网络。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/8d779ab7-ffcc-47c6-90ec-ede8260b2368.png)

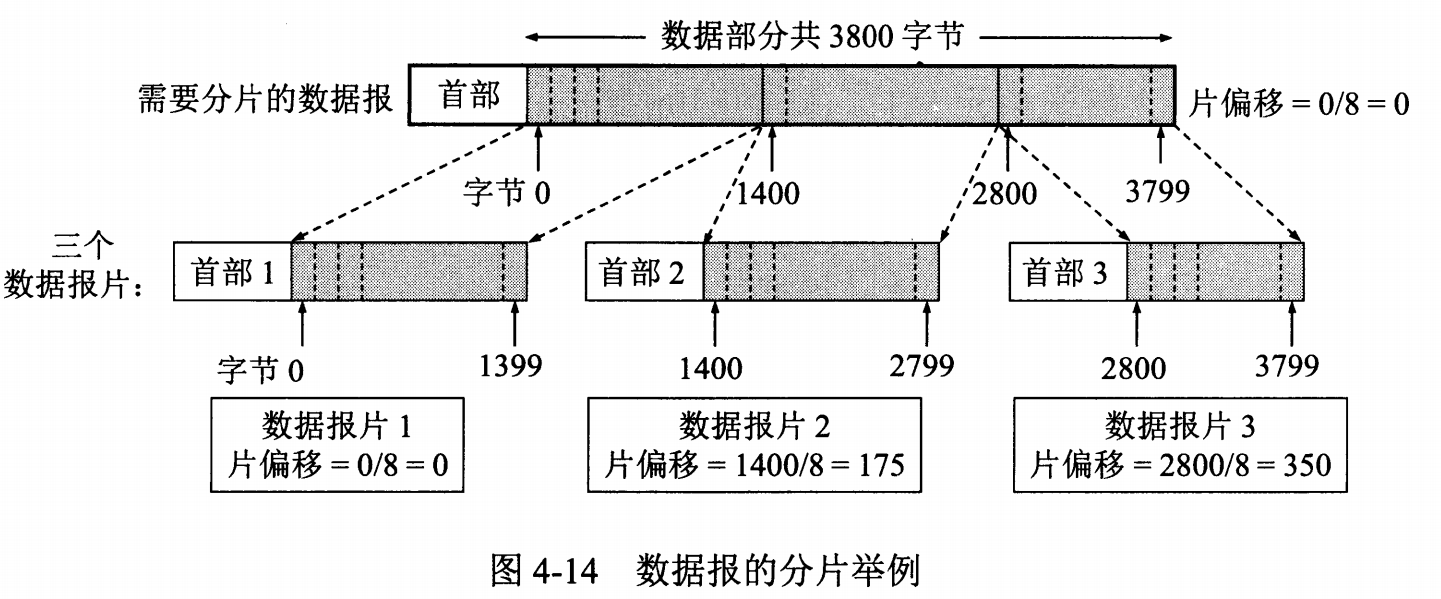
与 IP 协议配套使用的还有三个协议：

* 地址解析协议 ARP（Address Resolution Protocol）
* 网际控制报文协议 ICMP（Internet Control Message Protocol）
* 网际组管理协议 IGMP（Internet Group Management Protocol）

**IP 数据报格式**

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/85c05fb1-5546-4c50-9221-21f231cdc8c5.jpg)

* **版本** : 有 4（IPv4）和 6（IPv6）两个值；
* **首部长度** : 占 4 位，因此最大值为 15。值为 1 表示的是 1 个 32 位字的长度，也就是 4 字节。因为固定部分长度为 20 字节，因此该值最小为 5。如果可选字段的长度不是 4 字节的整数倍，就用尾部的填充部分来填充。
* **区分服务** : 用来获得更好的服务，一般情况下不使用。
* **总长度** : 包括首部长度和数据部分长度。
* **生存时间** ：TTL，它的存在是为了防止无法交付的数据报在互联网中不断兜圈子。以路由器跳数为单位，当 TTL 为 0 时就丢弃数据报。
* **协议** ：指出携带的数据应该上交给哪个协议进行处理，例如 ICMP、TCP、UDP 等。
* **首部检验和** ：因为数据报每经过一个路由器，都要重新计算检验和，因此检验和不包含数据部分可以减少计算的工作量。
* **标识** : 在数据报长度过长从而发生分片的情况下，相同数据报的不同分片具有相同的标识符。
* **片偏移** : 和标识符一起，用于发生分片的情况。片偏移的单位为 8 字节。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/23ba890e-e11c-45e2-a20c-64d217f83430.png)

**IP 地址编址方式**

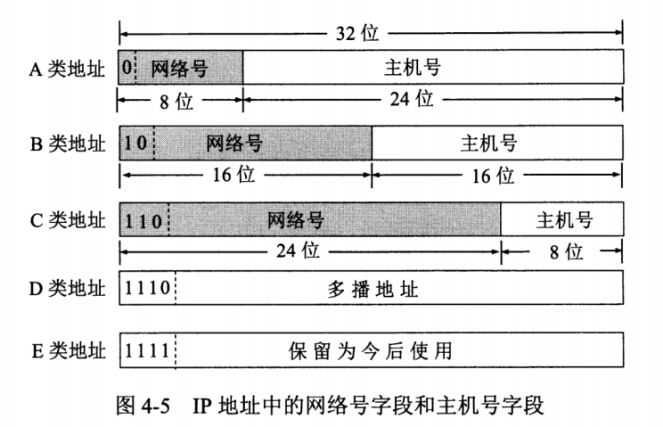
IP 地址的编址方式经历了三个历史阶段：

* 分类
* 子网划分
* 无分类

**1. 分类**

由两部分组成，网络号和主机号，其中不同分类具有不同的网络号长度，并且是固定的。

IP 地址 ::= {< 网络号 >, < 主机号 >}

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/cbf50eb8-22b4-4528-a2e7-d187143d57f7.png)

**2. 子网划分**

通过在主机号字段中拿一部分作为子网号，把两级 IP 地址划分为三级 IP 地址。

IP 地址 ::= {< 网络号 >, < 子网号 >, < 主机号 >}

要使用子网，必须配置子网掩码。一个 B 类地址的默认子网掩码为 255.255.0.0，如果 B 类地址的子网占两个比特，那么子网掩码为 11111111 11111111 11000000 00000000，也就是 255.255.192.0。

注意，外部网络看不到子网的存在。

**3. 无分类**

无分类编址 CIDR 消除了传统 A 类、B 类和 C 类地址以及划分子网的概念，使用网络前缀和主机号来对 IP 地址进行编码，网络前缀的长度可以根据需要变化。

IP 地址 ::= {< 网络前缀号 >, < 主机号 >}

CIDR 的记法上采用在 IP 地址后面加上网络前缀长度的方法，例如 128.14.35.7/20 表示前 20 位为网络前缀。

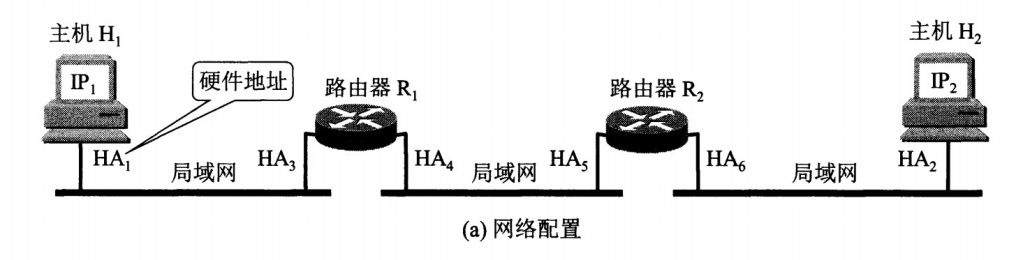
CIDR 的地址掩码可以继续称为子网掩码，子网掩码首 1 长度为网络前缀的长度。

一个 CIDR 地址块中有很多地址，一个 CIDR 表示的网络就可以表示原来的很多个网络，并且在路由表中只需要一个路由就可以代替原来的多个路由，减少了路由表项的数量。把这种通过使用网络前缀来减少路由表项的方式称为路由聚合，也称为**构成超网** 。

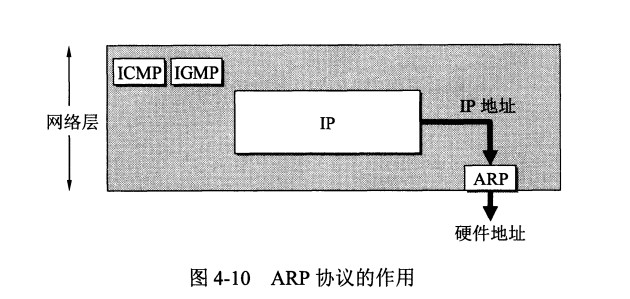
在路由表中的项目由“网络前缀”和“下一跳地址”组成，在查找时可能会得到不止一个匹配结果，应当采用最长前缀匹配来确定应该匹配哪一个。

**地址解析协议 ARP**

网络层实现主机之间的通信，而链路层实现具体每段链路之间的通信。因此在通信过程中，IP 数据报的源地址和目的地址始终不变，而 MAC 地址随着链路的改变而改变。

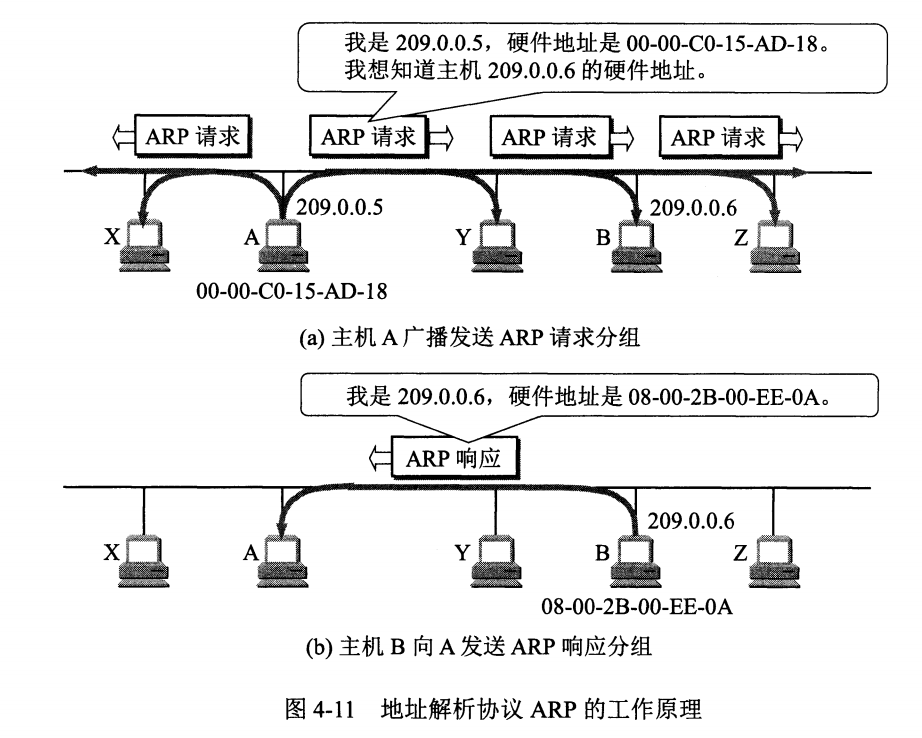
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/66192382-558b-4b05-a35d-ac4a2b1a9811.jpg)

ARP 实现由 IP 地址得到 MAC 地址。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/b9d79a5a-e7af-499b-b989-f10483e71b8b.jpg)

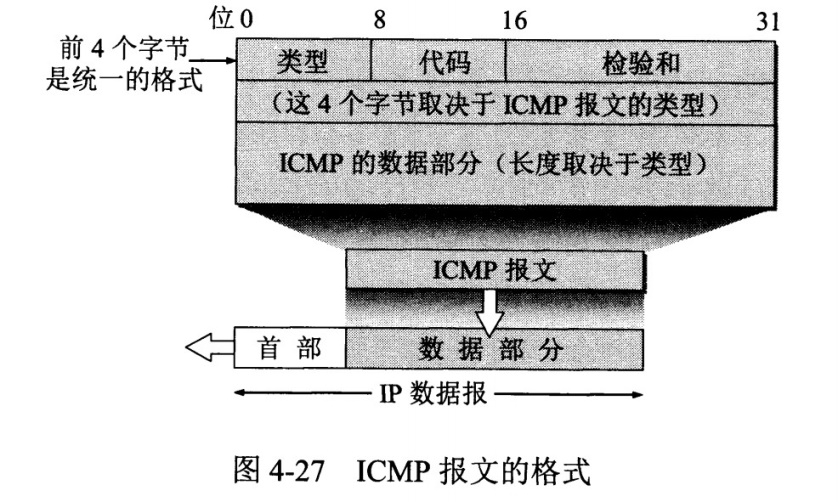
每个主机都有一个 ARP 高速缓存，里面有本局域网上的各主机和路由器的 IP 地址到 MAC 地址的映射表。

如果主机 A 知道主机 B 的 IP 地址，但是 ARP 高速缓存中没有该 IP 地址到 MAC 地址的映射，此时主机 A 通过广播的方式发送 ARP 请求分组，主机 B 收到该请求后会发送 ARP 响应分组给主机 A 告知其 MAC 地址，随后主机 A 向其高速缓存中写入主机 B 的 IP 地址到 MAC 地址的映射。

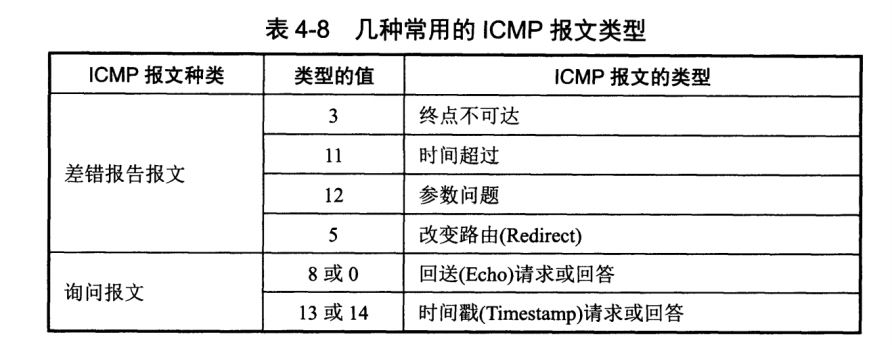
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/8006a450-6c2f-498c-a928-c927f758b1d0.png)

**网际控制报文协议 ICMP**

ICMP 是为了更有效地转发 IP 数据报和提高交付成功的机会。它封装在 IP 数据报中，但是不属于高层协议。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/e3124763-f75e-46c3-ba82-341e6c98d862.jpg)

ICMP 报文分为差错报告报文和询问报文。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/aa29cc88-7256-4399-8c7f-3cf4a6489559.png)

**1. Ping**

Ping 是 ICMP 的一个重要应用，主要用来测试两台主机之间的连通性。

Ping 的原理是通过向目的主机发送 ICMP Echo 请求报文，目的主机收到之后会发送 Echo 回答报文。Ping 会根据时间和成功响应的次数估算出数据包往返时间以及丢包率。

**2. Traceroute**

Traceroute 是 ICMP 的另一个应用，用来跟踪一个分组从源点到终点的路径。

Traceroute 发送的 IP 数据报封装的是无法交付的 UDP 用户数据报，并由目的主机发送终点不可达差错报告报文。

* 源主机向目的主机发送一连串的 IP 数据报。第一个数据报 P1 的生存时间 TTL 设置为 1，当 P1 到达路径上的第一个路由器 R1 时，R1 收下它并把 TTL 减 1，此时 TTL 等于 0，R1 就把 P1 丢弃，并向源主机发送一个 ICMP 时间超过差错报告报文；
* 源主机接着发送第二个数据报 P2，并把 TTL 设置为 2。P2 先到达 R1，R1 收下后把 TTL 减 1 再转发给 R2，R2 收下后也把 TTL 减 1，由于此时 TTL 等于 0，R2 就丢弃 P2，并向源主机发送一个 ICMP 时间超过差错报文。
* 不断执行这样的步骤，直到最后一个数据报刚刚到达目的主机，主机不转发数据报，也不把 TTL 值减 1。但是因为数据报封装的是无法交付的 UDP，因此目的主机要向源主机发送 ICMP 终点不可达差错报告报文。
* 之后源主机知道了到达目的主机所经过的路由器 IP 地址以及到达每个路由器的往返时间。

**虚拟专用网 VPN**

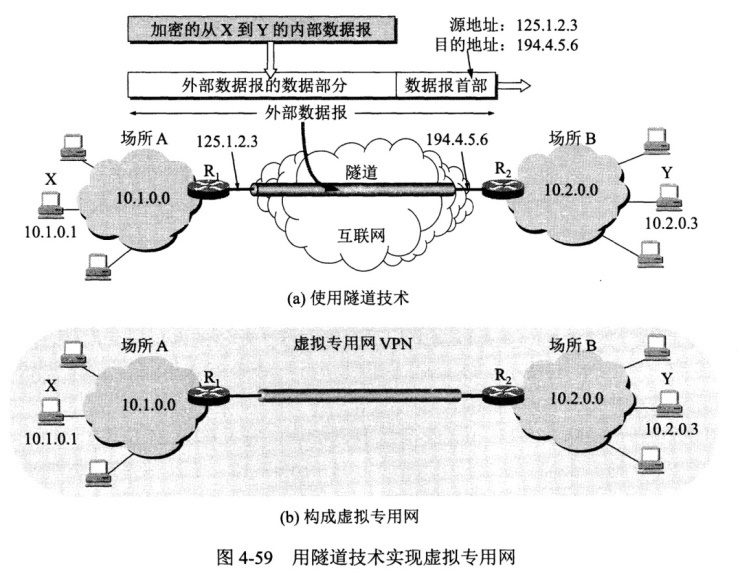
由于 IP 地址的紧缺，一个机构能申请到的 IP 地址数往往远小于本机构所拥有的主机数。并且一个机构并不需要把所有的主机接入到外部的互联网中，机构内的计算机可以使用仅在本机构有效的 IP 地址（专用地址）。

有三个专用地址块：

* 10.0.0.0 ~ 10.255.255.255
* 172.16.0.0 ~ 172.31.255.255
* 192.168.0.0 ~ 192.168.255.255

VPN 使用公用的互联网作为本机构各专用网之间的通信载体。专用指机构内的主机只与本机构内的其它主机通信；虚拟指好像是，而实际上并不是，它有经过公用的互联网。

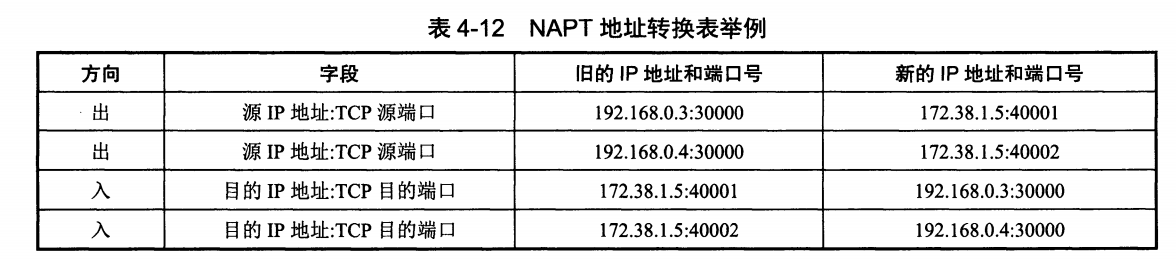
下图中，场所 A 和 B 的通信经过互联网，如果场所 A 的主机 X 要和另一个场所 B 的主机 Y 通信，IP 数据报的源地址是 10.1.0.1，目的地址是 10.2.0.3。数据报先发送到与互联网相连的路由器 R1，R1 对内部数据进行加密，然后重新加上数据报的首部，源地址是路由器 R1 的全球地址 125.1.2.3，目的地址是路由器 R2 的全球地址 194.4.5.6。路由器 R2 收到数据报后将数据部分进行解密，恢复原来的数据报，此时目的地址为 10.2.0.3，就交付给 Y。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/1556770b-8c01-4681-af10-46f1df69202c.jpg)

**网络地址转换 NAT**

专用网内部的主机使用本地 IP 地址又想和互联网上的主机通信时，可以使用 NAT 来将本地 IP 转换为全球 IP。

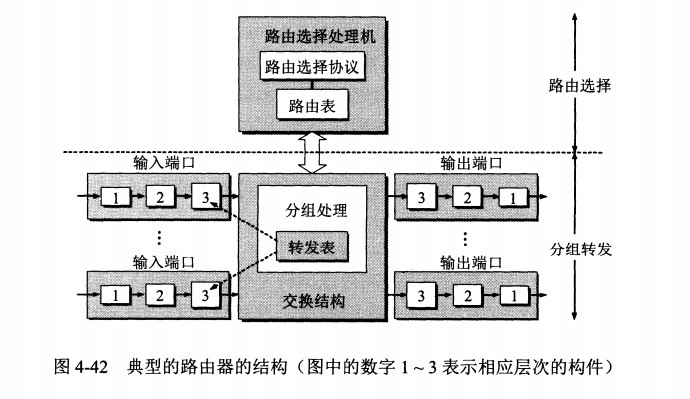
在以前，NAT 将本地 IP 和全球 IP 一一对应，这种方式下拥有 n 个全球 IP 地址的专用网内最多只可以同时有 n 台主机接入互联网。为了更有效地利用全球 IP 地址，现在常用的 NAT 转换表把传输层的端口号也用上了，使得多个专用网内部的主机共用一个全球 IP 地址。使用端口号的 NAT 也叫做网络地址与端口转换 NAPT。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/2719067e-b299-4639-9065-bed6729dbf0b.png)

**路由器的结构**

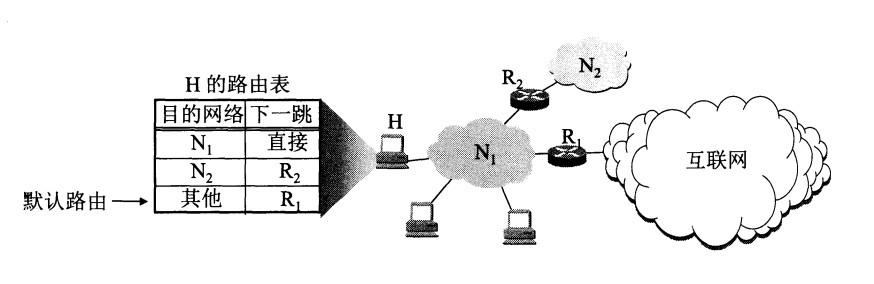
路由器从功能上可以划分为：路由选择和分组转发。

分组转发结构由三个部分组成：交换结构、一组输入端口和一组输出端口。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/c3369072-c740-43b0-b276-202bd1d3960d.jpg)

**路由器分组转发流程**

* 从数据报的首部提取目的主机的 IP 地址 D，得到目的网络地址 N。
* 若 N 就是与此路由器直接相连的某个网络地址，则进行直接交付；
* 若路由表中有目的地址为 D 的特定主机路由，则把数据报传送给表中所指明的下一跳路由器；
* 若路由表中有到达网络 N 的路由，则把数据报传送给路由表中所指明的下一跳路由器；
* 若路由表中有一个默认路由，则把数据报传送给路由表中所指明的默认路由器；
* 报告转发分组出错。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/1ab49e39-012b-4383-8284-26570987e3c4.jpg)

**路由选择协议**

路由选择协议都是自适应的，能随着网络通信量和拓扑结构的变化而自适应地进行调整。

互联网可以划分为许多较小的自治系统 AS，一个 AS 可以使用一种和别的 AS 不同的路由选择协议。

可以把路由选择协议划分为两大类：

* 自治系统内部的路由选择：RIP 和 OSPF
* 自治系统间的路由选择：BGP

**1. 内部网关协议 RIP**

RIP 是一种基于距离向量的路由选择协议。距离是指跳数，直接相连的路由器跳数为 1。跳数最多为 15，超过 15 表示不可达。

RIP 按固定的时间间隔仅和相邻路由器交换自己的路由表，经过若干次交换之后，所有路由器最终会知道到达本自治系统中任何一个网络的最短距离和下一跳路由器地址。

距离向量算法：

* 对地址为 X 的相邻路由器发来的 RIP 报文，先修改报文中的所有项目，把下一跳字段中的地址改为 X，并把所有的距离字段加 1；
* 对修改后的 RIP 报文中的每一个项目，进行以下步骤：
* 若原来的路由表中没有目的网络 N，则把该项目添加到路由表中；
* 否则：若下一跳路由器地址是 X，则把收到的项目替换原来路由表中的项目；否则：若收到的项目中的距离 d 小于路由表中的距离，则进行更新（例如原始路由表项为 Net2, 5, P，新表项为 Net2, 4, X，则更新）；否则什么也不做。
* 若 3 分钟还没有收到相邻路由器的更新路由表，则把该相邻路由器标为不可达，即把距离置为 16。

RIP 协议实现简单，开销小。但是 RIP 能使用的最大距离为 15，限制了网络的规模。并且当网络出现故障时，要经过比较长的时间才能将此消息传送到所有路由器。

**2. 内部网关协议 OSPF**

开放最短路径优先 OSPF，是为了克服 RIP 的缺点而开发出来的。

开放表示 OSPF 不受某一家厂商控制，而是公开发表的；最短路径优先表示使用了 Dijkstra 提出的最短路径算法 SPF。

OSPF 具有以下特点：

* 向本自治系统中的所有路由器发送信息，这种方法是洪泛法。
* 发送的信息就是与相邻路由器的链路状态，链路状态包括与哪些路由器相连以及链路的度量，度量用费用、距离、时延、带宽等来表示。
* 只有当链路状态发生变化时，路由器才会发送信息。

所有路由器都具有全网的拓扑结构图，并且是一致的。相比于 RIP，OSPF 的更新过程收敛的很快。

**3. 外部网关协议 BGP**

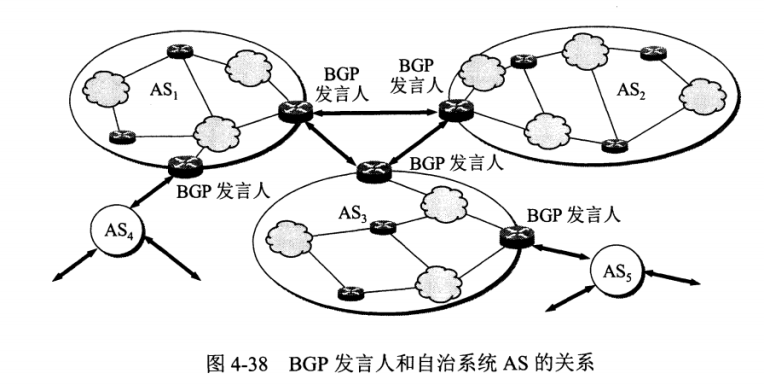
BGP（Border Gateway Protocol，边界网关协议）

AS 之间的路由选择很困难，主要是由于：

* 互联网规模很大；
* 各个 AS 内部使用不同的路由选择协议，无法准确定义路径的度量；
* AS 之间的路由选择必须考虑有关的策略，比如有些 AS 不愿意让其它 AS 经过。

BGP 只能寻找一条比较好的路由，而不是最佳路由。

每个 AS 都必须配置 BGP 发言人，通过在两个相邻 BGP 发言人之间建立 TCP 连接来交换路由信息。



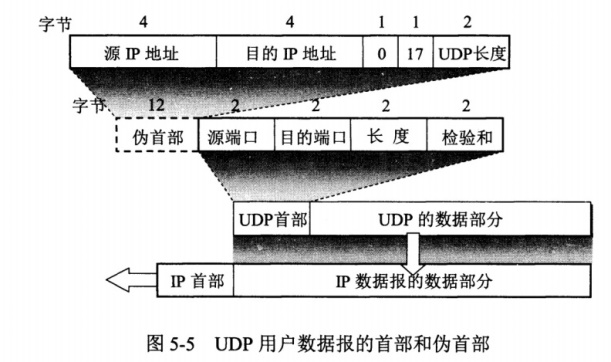
传输层

网络层只把分组发送到目的主机，但是真正通信的并不是主机而是主机中的进程。传输层提供了进程间的逻辑通信，传输层向高层用户屏蔽了下面网络层的核心细节，使应用程序看起来像是在两个传输层实体之间有一条端到端的逻辑通信信道。

**UDP 和 TCP 的特点**

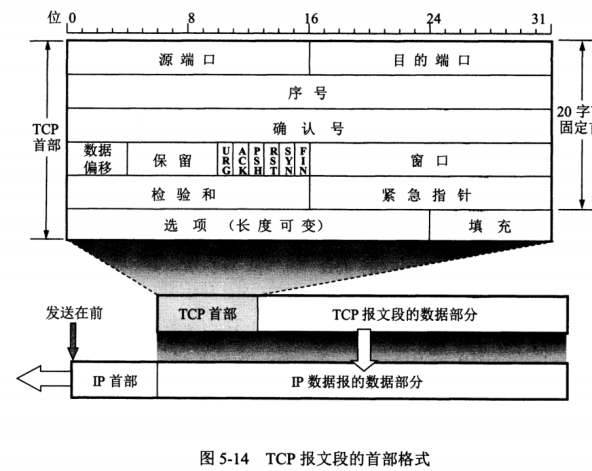
* 用户数据报协议 UDP（User Datagram Protocol）是无连接的，尽最大可能交付，没有拥塞控制，面向报文（对于应用程序传下来的报文不合并也不拆分，只是添加 UDP 首部），支持一对一、一对多、多对一和多对多的交互通信。
* 传输控制协议 TCP（Transmission Control Protocol）是面向连接的，提供可靠交付，有流量控制，拥塞控制，提供全双工通信，面向字节流（把应用层传下来的报文看成字节流，把字节流组织成大小不等的数据块），每一条 TCP 连接只能是点对点的（一对一）。

**UDP 首部格式**

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/d4c3a4a1-0846-46ec-9cc3-eaddfca71254.jpg)

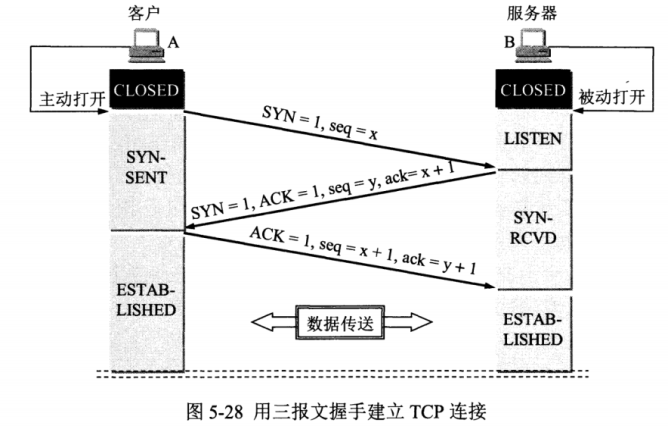
首部字段只有 8 个字节，包括源端口、目的端口、长度、检验和。12 字节的伪首部是为了计算检验和临时添加的。

**TCP 首部格式**

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/55dc4e84-573d-4c13-a765-52ed1dd251f9.png)

* **序号** ：用于对字节流进行编号，例如序号为 301，表示第一个字节的编号为 301，如果携带的数据长度为 100 字节，那么下一个报文段的序号应为 401。
* **确认号** ：期望收到的下一个报文段的序号。例如 B 正确收到 A 发送来的一个报文段，序号为 501，携带的数据长度为 200 字节，因此 B 期望下一个报文段的序号为 701，B 发送给 A 的确认报文段中确认号就为 701。
* **数据偏移** ：指的是数据部分距离报文段起始处的偏移量，实际上指的是首部的长度。
* **确认 ACK** ：当 ACK=1 时确认号字段有效，否则无效。TCP 规定，在连接建立后所有传送的报文段都必须把 ACK 置 1。
* **同步 SYN** ：在连接建立时用来同步序号。当 SYN=1，ACK=0 时表示这是一个连接请求报文段。若对方同意建立连接，则响应报文中 SYN=1，ACK=1。
* **终止 FIN** ：用来释放一个连接，当 FIN=1 时，表示此报文段的发送方的数据已发送完毕，并要求释放连接。
* **窗口** ：窗口值作为接收方让发送方设置其发送窗口的依据。之所以要有这个限制，是因为接收方的数据缓存空间是有限的。

**TCP 的三次握手**

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/e92d0ebc-7d46-413b-aec1-34a39602f787.png)

假设 A 为客户端，B 为服务器端。

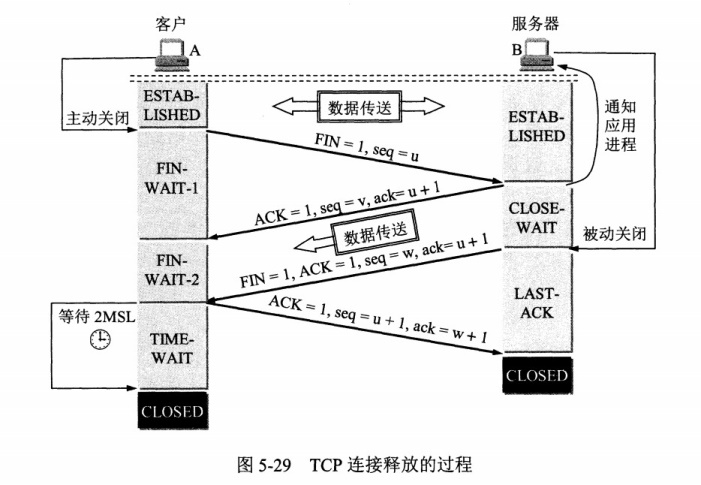
* 首先 B 处于 LISTEN（监听）状态，等待客户的连接请求。
* A 向 B 发送连接请求报文，SYN=1，ACK=0，选择一个初始的序号 x。
* B 收到连接请求报文，如果同意建立连接，则向 A 发送连接确认报文，SYN=1，ACK=1，确认号为 x+1，同时也选择一个初始的序号 y。
* A 收到 B 的连接确认报文后，还要向 B 发出确认，确认号为 y+1，序号为 x+1。
* B 收到 A 的确认后，连接建立。

**三次握手的原因**

第三次握手是为了防止失效的连接请求到达服务器，让服务器错误打开连接。

客户端发送的连接请求如果在网络中滞留，那么就会隔很长一段时间才能收到服务器端发回的连接确认。客户端等待一个超时重传时间之后，就会重新请求连接。但是这个滞留的连接请求最后还是会到达服务器，如果不进行三次握手，那么服务器就会打开两个连接。如果有第三次握手，客户端会忽略服务器之后发送的对滞留连接请求的连接确认，不进行第三次握手，因此就不会再次打开连接。

**TCP 的四次挥手**

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/f87afe72-c2df-4c12-ac03-9b8d581a8af8.jpg)

以下描述不讨论序号和确认号，因为序号和确认号的规则比较简单。并且不讨论 ACK，因为 ACK 在连接建立之后都为 1。

* A 发送连接释放报文，FIN=1。
* B 收到之后发出确认，此时 TCP 属于半关闭状态，B 能向 A 发送数据但是 A 不能向 B 发送数据。
* 当 B 不再需要连接时，发送连接释放报文，FIN=1。
* A 收到后发出确认，进入 TIME-WAIT 状态，等待 2 MSL（最大报文存活时间）后释放连接。
* B 收到 A 的确认后释放连接。

**四次挥手的原因**

客户端发送了 FIN 连接释放报文之后，服务器收到了这个报文，就进入了 CLOSE-WAIT 状态。这个状态是为了让服务器端发送还未传送完毕的数据，传送完毕之后，服务器会发送 FIN 连接释放报文。

**TIME\_WAIT**

客户端接收到服务器端的 FIN 报文后进入此状态，此时并不是直接进入 CLOSED 状态，还需要等待一个时间计时器设置的时间 2MSL。这么做有两个理由：

* 确保最后一个确认报文能够到达。如果 B 没收到 A 发送来的确认报文，那么就会重新发送连接释放请求报文，A 等待一段时间就是为了处理这种情况的发生。
* 等待一段时间是为了让本连接持续时间内所产生的所有报文都从网络中消失，使得下一个新的连接不会出现旧的连接请求报文。

**TCP 可靠传输**

TCP 使用超时重传来实现可靠传输：如果一个已经发送的报文段在超时时间内没有收到确认，那么就重传这个报文段。

一个报文段从发送再到接收到确认所经过的时间称为往返时间 RTT，加权平均往返时间 RTTs 计算如下：

[https://camo.githubusercontent.com/02e9a4a23b05bcf814d9e5d0c1b79a6bb5ae2aee/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f525454733d28312d61292a2852545473292b612a525454](https://camo.githubusercontent.com/02e9a4a23b05bcf814d9e5d0c1b79a6bb5ae2aee/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f525454733d28312d61292a2852545473292b612a525454)

其中，0 ≤ a ＜ 1，RTTs 随着 a 的增加更容易受到 RTT 的影响。

超时时间 RTO 应该略大于 RTTs，TCP 使用的超时时间计算如下：

[https://camo.githubusercontent.com/da3da0a6cad7ef2af7d7adcd2940277780c35325/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f52544f3d525454732b342a5254545f64](https://camo.githubusercontent.com/da3da0a6cad7ef2af7d7adcd2940277780c35325/68747470733a2f2f6c617465782e636f6465636f67732e636f6d2f6769662e6c617465783f52544f3d525454732b342a5254545f64)

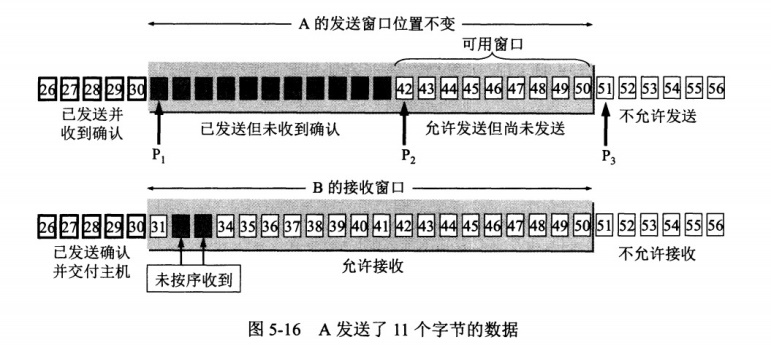
其中 RTTd 为偏差的加权平均值。

**TCP 滑动窗口**

窗口是缓存的一部分，用来暂时存放字节流。发送方和接收方各有一个窗口，接收方通过 TCP 报文段中的窗口字段告诉发送方自己的窗口大小，发送方根据这个值和其它信息设置自己的窗口大小。

发送窗口内的字节都允许被发送，接收窗口内的字节都允许被接收。如果发送窗口左部的字节已经发送并且收到了确认，那么就将发送窗口向右滑动一定距离，直到左部第一个字节不是已发送并且已确认的状态；接收窗口的滑动类似，接收窗口左部字节已经发送确认并交付主机，就向右滑动接收窗口。

接收窗口只会对窗口内最后一个按序到达的字节进行确认，例如接收窗口已经收到的字节为 {31, 34, 35}，其中 {31} 按序到达，而 {34, 35} 就不是，因此只对字节 31 进行确认。发送方得到一个字节的确认之后，就知道这个字节之前的所有字节都已经被接收。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/a3253deb-8d21-40a1-aae4-7d178e4aa319.jpg)

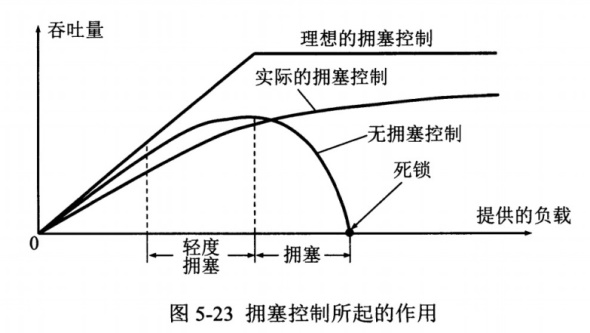
**TCP 流量控制**

流量控制是为了控制发送方发送速率，保证接收方来得及接收。

接收方发送的确认报文中的窗口字段可以用来控制发送方窗口大小，从而影响发送方的发送速率。将窗口字段设置为 0，则发送方不能发送数据。

**TCP 拥塞控制**

如果网络出现拥塞，分组将会丢失，此时发送方会继续重传，从而导致网络拥塞程度更高。因此当出现拥塞时，应当控制发送方的速率。这一点和流量控制很像，但是出发点不同。流量控制是为了让接收方能来得及接收，而拥塞控制是为了降低整个网络的拥塞程度。

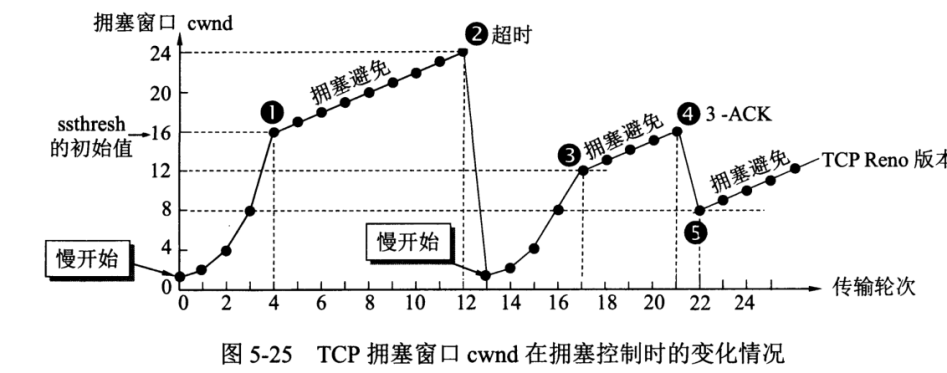
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/51e2ed95-65b8-4ae9-8af3-65602d452a25.jpg)

TCP 主要通过四个算法来进行拥塞控制：慢开始、拥塞避免、快重传、快恢复。

发送方需要维护一个叫做拥塞窗口（cwnd）的状态变量，注意拥塞窗口与发送方窗口的区别：拥塞窗口只是一个状态变量，实际决定发送方能发送多少数据的是发送方窗口。

为了便于讨论，做如下假设：

* 接收方有足够大的接收缓存，因此不会发生流量控制；
* 虽然 TCP 的窗口基于字节，但是这里设窗口的大小单位为报文段。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/910f613f-514f-4534-87dd-9b4699d59d31.png)

**1. 慢开始与拥塞避免**

发送的最初执行慢开始，令 cwnd = 1，发送方只能发送 1 个报文段；当收到确认后，将 cwnd 加倍，因此之后发送方能够发送的报文段数量为：2、4、8 ...

注意到慢开始每个轮次都将 cwnd 加倍，这样会让 cwnd 增长速度非常快，从而使得发送方发送的速度增长速度过快，网络拥塞的可能性也就更高。设置一个慢开始门限 ssthresh，当 cwnd >= ssthresh 时，进入拥塞避免，每个轮次只将 cwnd 加 1。

如果出现了超时，则令 ssthresh = cwnd / 2，然后重新执行慢开始。

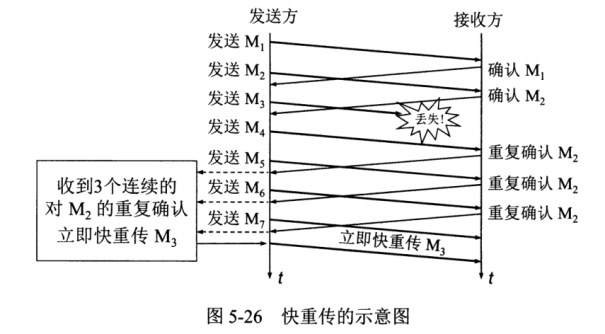
**2. 快重传与快恢复**

在接收方，要求每次接收到报文段都应该对最后一个已收到的有序报文段进行确认。例如已经接收到 M1 和 M2，此时收到 M4，应当发送对 M2 的确认。

在发送方，如果收到三个重复确认，那么可以知道下一个报文段丢失，此时执行快重传，立即重传下一个报文段。例如收到三个 M2，则 M3 丢失，立即重传 M3。

在这种情况下，只是丢失个别报文段，而不是网络拥塞。因此执行快恢复，令 ssthresh = cwnd / 2 ，cwnd = ssthresh，注意到此时直接进入拥塞避免。

慢开始和快恢复的快慢指的是 cwnd 的设定值，而不是 cwnd 的增长速率。慢开始 cwnd 设定为 1，而快恢复 cwnd 设定为 ssthresh。

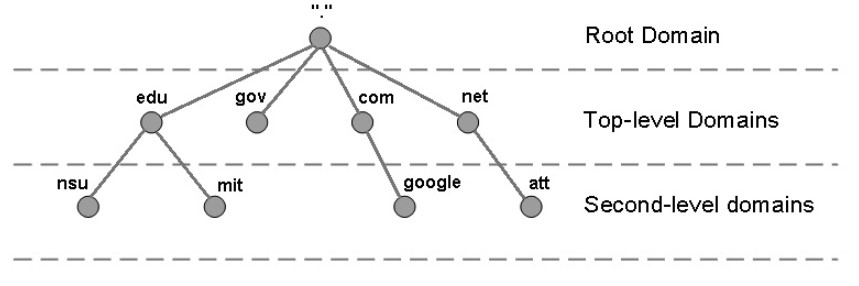
[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/f61b5419-c94a-4df1-8d4d-aed9ae8cc6d5.png)

应用层

**域名系统**

DNS 是一个分布式数据库，提供了主机名和 IP 地址之间相互转换的服务。这里的分布式数据库是指，每个站点只保留它自己的那部分数据。

域名具有层次结构，从上到下依次为：根域名、顶级域名、二级域名。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/b54eeb16-0b0e-484c-be62-306f57c40d77.jpg)

DNS 可以使用 UDP 或者 TCP 进行传输，使用的端口号都为 53。大多数情况下 DNS 使用 UDP 进行传输，这就要求域名解析器和域名服务器都必须自己处理超时和重传从而保证可靠性。在两种情况下会使用 TCP 进行传输：

* 如果返回的响应超过的 512 字节（UDP 最大只支持 512 字节的数据）。
* 区域传送（区域传送是主域名服务器向辅助域名服务器传送变化的那部分数据）。

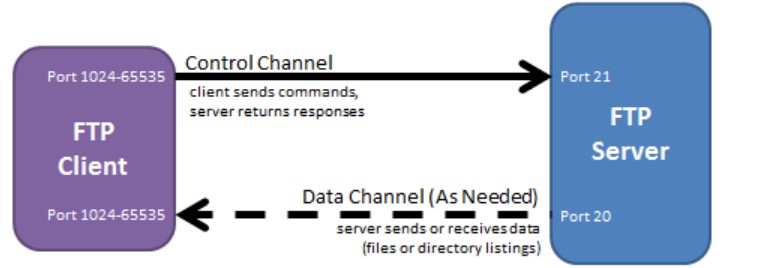
**文件传送协议**

FTP 使用 TCP 进行连接，它需要两个连接来传送一个文件：

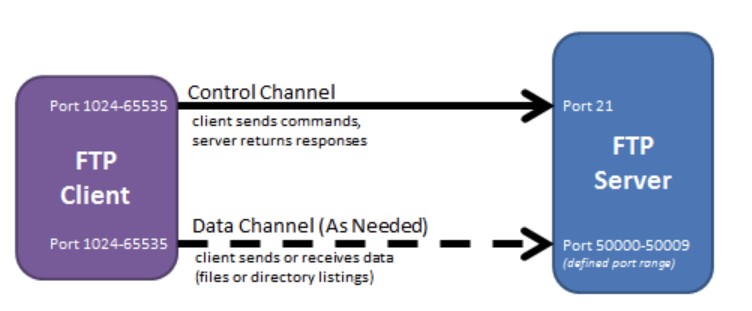
* 控制连接：服务器打开端口号 21 等待客户端的连接，客户端主动建立连接后，使用这个连接将客户端的命令传送给服务器，并传回服务器的应答。
* 数据连接：用来传送一个文件数据。

根据数据连接是否是服务器端主动建立，FTP 有主动和被动两种模式：

* 主动模式：服务器端主动建立数据连接，其中服务器端的端口号为 20，客户端的端口号随机，但是必须大于 1024，因为 0~1023 是熟知端口号。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/03f47940-3843-4b51-9e42-5dcaff44858b.jpg)

* 被动模式：客户端主动建立数据连接，其中客户端的端口号由客户端自己指定，服务器端的端口号随机。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/be5c2c61-86d2-4dba-a289-b48ea23219de.jpg)

主动模式要求客户端开放端口号给服务器端，需要去配置客户端的防火墙。被动模式只需要服务器端开放端口号即可，无需客户端配置防火墙。但是被动模式会导致服务器端的安全性减弱，因为开放了过多的端口号。

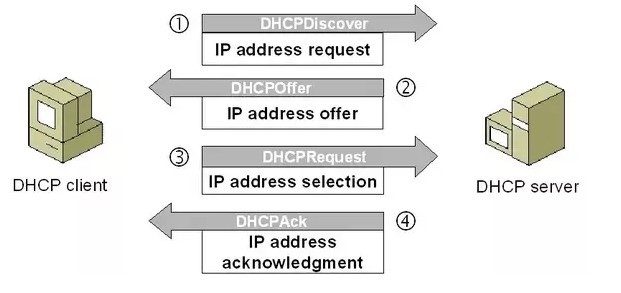
**动态主机配置协议**

DHCP (Dynamic Host Configuration Protocol) 提供了即插即用的连网方式，用户不再需要手动配置 IP 地址等信息。

DHCP 配置的内容不仅是 IP 地址，还包括子网掩码、网关 IP 地址。

DHCP 工作过程如下：

1. 客户端发送 Discover 报文，该报文的目的地址为 255.255.255.255:67，源地址为 0.0.0.0:68，被放入 UDP 中，该报文被广播到同一个子网的所有主机上。如果客户端和 DHCP 服务器不在同一个子网，就需要使用中继代理。
2. DHCP 服务器收到 Discover 报文之后，发送 Offer 报文给客户端，该报文包含了客户端所需要的信息。因为客户端可能收到多个 DHCP 服务器提供的信息，因此客户端需要进行选择。
3. 如果客户端选择了某个 DHCP 服务器提供的信息，那么就发送 Request 报文给该 DHCP 服务器。
4. DHCP 服务器发送 Ack 报文，表示客户端此时可以使用提供给它的信息。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/23219e4c-9fc0-4051-b33a-2bd95bf054ab.jpg)

**远程登录协议**

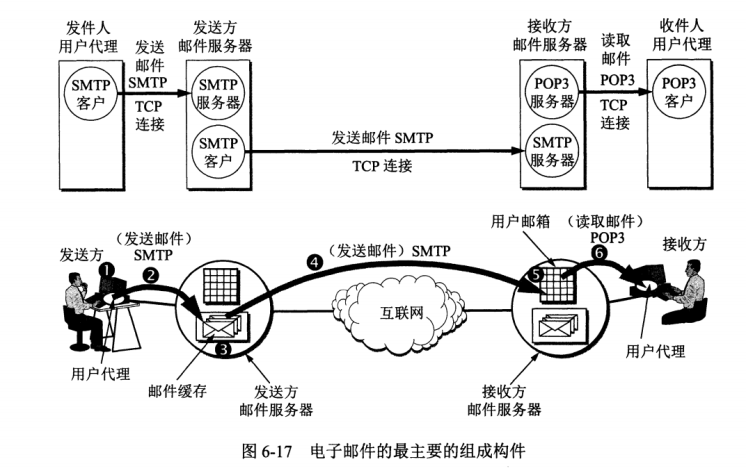
TELNET 用于登录到远程主机上，并且远程主机上的输出也会返回。

TELNET 可以适应许多计算机和操作系统的差异，例如不同操作系统系统的换行符定义。

**电子邮件协议**

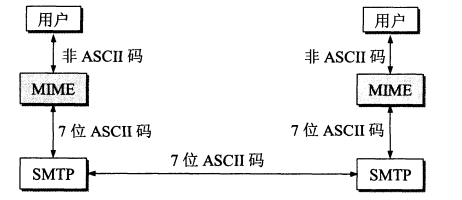
一个电子邮件系统由三部分组成：用户代理、邮件服务器以及邮件协议。

邮件协议包含发送协议和读取协议，发送协议常用 SMTP，读取协议常用 POP3 和 IMAP。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/7b3efa99-d306-4982-8cfb-e7153c33aab4.png)

**1. SMTP**

SMTP 只能发送 ASCII 码，而互联网邮件扩充 MIME 可以发送二进制文件。MIME 并没有改动或者取代 SMTP，而是增加邮件主体的结构，定义了非 ASCII 码的编码规则。

[](https://github.com/CyC2018/CS-Notes/blob/master/notes/pics/ed5522bb-3a60-481c-8654-43e7195a48fe.png)

**2. POP3**

POP3 的特点是只要用户从服务器上读取了邮件，就把该邮件删除。但最新版本的 POP3 可以不删除邮件。

**3. IMAP**

IMAP 协议中客户端和服务器上的邮件保持同步，如果不手动删除邮件，那么服务器上的邮件也不会被删除。IMAP 这种做法可以让用户随时随地去访问服务器上的邮件。

**常用端口**

| **应用** | **应用层协议** | **端口号** | **传输层协议** | **备注** |
| --- | --- | --- | --- | --- |
| 域名解析 | DNS | 53 | UDP/TCP | 长度超过 512 字节时使用 TCP |
| 动态主机配置协议 | DHCP | 67/68 | UDP |  |
| 简单网络管理协议 | SNMP | 161/162 | UDP |  |
| 文件传送协议 | FTP | 20/21 | TCP | 控制连接 21，数据连接 20 |
| 远程终端协议 | TELNET | 23 | TCP |  |
| 超文本传送协议 | HTTP | 80 | TCP |  |
| 简单邮件传送协议 | SMTP | 25 | TCP |  |
| 邮件读取协议 | POP3 | 110 | TCP |  |
| 网际报文存取协议 | IMAP | 143 | TCP |  |

**Web 页面请求过程**

**1. DHCP 配置主机信息**

* 假设主机最开始没有 IP 地址以及其它信息，那么就需要先使用 DHCP 来获取。
* 主机生成一个 DHCP 请求报文，并将这个报文放入具有目的端口 67 和源端口 68 的 UDP 报文段中。
* 该报文段则被放入在一个具有广播 IP 目的地址(255.255.255.255) 和源 IP 地址（0.0.0.0）的 IP 数据报中。
* 该数据报则被放置在 MAC 帧中，该帧具有目的地址 FF:FF:FF:FF:FF:FF，将广播到与交换机连接的所有设备。
* 连接在交换机的 DHCP 服务器收到广播帧之后，不断地向上分解得到 IP 数据报、UDP 报文段、DHCP 请求报文，之后生成 DHCP ACK 报文，该报文包含以下信息：IP 地址、DNS 服务器的 IP 地址、默认网关路由器的 IP 地址和子网掩码。该报文被放入 UDP 报文段中，UDP 报文段有被放入 IP 数据报中，最后放入 MAC 帧中。
* 该帧的目的地址是请求主机的 MAC 地址，因为交换机具有自学习能力，之前主机发送了广播帧之后就记录了 MAC 地址到其转发接口的交换表项，因此现在交换机就可以直接知道应该向哪个接口发送该帧。
* 主机收到该帧后，不断分解得到 DHCP 报文。之后就配置它的 IP 地址、子网掩码和 DNS 服务器的 IP 地址，并在其 IP 转发表中安装默认网关。

**2. ARP 解析 MAC 地址**

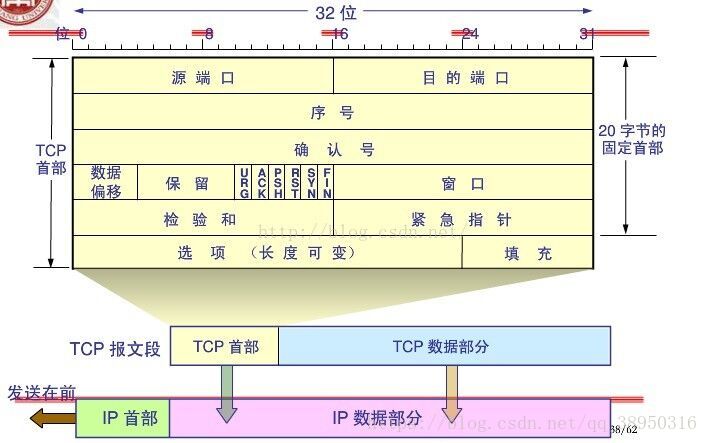
* 主机通过浏览器生成一个 TCP 套接字，套接字向 HTTP 服务器发送 HTTP 请求。为了生成该套接字，主机需要知道网站的域名对应的 IP 地址。
* 主机生成一个 DNS 查询报文，该报文具有 53 号端口，因为 DNS 服务器的端口号是 53。
* 该 DNS 查询报文被放入目的地址为 DNS 服务器 IP 地址的 IP 数据报中。
* 该 IP 数据报被放入一个以太网帧中，该帧将发送到网关路由器。
* DHCP 过程只知道网关路由器的 IP 地址，为了获取网关路由器的 MAC 地址，需要使用 ARP 协议。
* 主机生成一个包含目的地址为网关路由器 IP 地址的 ARP 查询报文，将该 ARP 查询报文放入一个具有广播目的地址（FF:FF:FF:FF:FF:FF）的以太网帧中，并向交换机发送该以太网帧，交换机将该帧转发给所有的连接设备，包括网关路由器。
* 网关路由器接收到该帧后，不断向上分解得到 ARP 报文，发现其中的 IP 地址与其接口的 IP 地址匹配，因此就发送一个 ARP 回答报文，包含了它的 MAC 地址，发回给主机。

**3. DNS 解析域名**

* 知道了网关路由器的 MAC 地址之后，就可以继续 DNS 的解析过程了。
* 网关路由器接收到包含 DNS 查询报文的以太网帧后，抽取出 IP 数据报，并根据转发表决定该 IP 数据报应该转发的路由器。
* 因为路由器具有内部网关协议（RIP、OSPF）和外部网关协议（BGP）这两种路由选择协议，因此路由表中已经配置了网关路由器到达 DNS 服务器的路由表项。
* 到达 DNS 服务器之后，DNS 服务器抽取出 DNS 查询报文，并在 DNS 数据库中查找待解析的域名。
* 找到 DNS 记录之后，发送 DNS 回答报文，将该回答报文放入 UDP 报文段中，然后放入 IP 数据报中，通过路由器反向转发回网关路由器，并经过以太网交换机到达主机。

**4. HTTP 请求页面**

* 有了 HTTP 服务器的 IP 地址之后，主机就能够生成 TCP 套接字，该套接字将用于向 Web 服务器发送 HTTP GET 报文。
* 在生成 TCP 套接字之前，必须先与 HTTP 服务器进行三次握手来建立连接。生成一个具有目的端口 80 的 TCP SYN 报文段，并向 HTTP 服务器发送该报文段。
* HTTP 服务器收到该报文段之后，生成 TCP SYN ACK 报文段，发回给主机。
* 连接建立之后，浏览器生成 HTTP GET 报文，并交付给 HTTP 服务器。
* HTTP 服务器从 TCP 套接字读取 HTTP GET 报文，生成一个 HTTP 响应报文，将 Web 页面内容放入报文主体中，发回给主机。
* 浏览器收到 HTTP 响应报文后，抽取出 Web 页面内容，之后进行渲染，显示 Web 页面。



    序列号seq：占4个字节，用来标记数据段的顺序，TCP把连接中发送的所有数据字节都编上一个序号，第一个字节的编号由本地随机产生；给字节编上序号后，就给每一个报文段指派一个序号；序列号seq就是这个报文段中的第一个字节的数据编号。

    确认号ack：占4个字节，期待收到对方下一个报文段的第一个数据字节的序号；序列号表示报文段携带数据的第一个字节的编号；而确认号指的是期望接收到下一个字节的编号；因此当前报文段最后一个字节的编号+1即为确认号。

    确认ACK：占1位，仅当ACK=1时，确认号字段才有效。ACK=0时，确认号无效

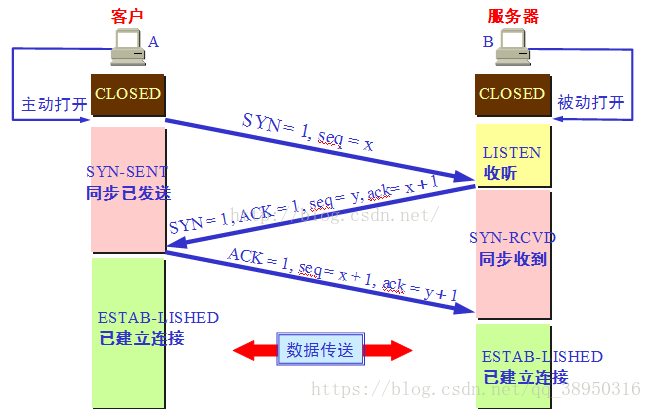
    同步SYN：连接建立时用于同步序号。当SYN=1，ACK=0时表示：这是一个连接请求报文段。若同意连接，则在响应报文段中使得SYN=1，ACK=1。因此，SYN=1表示这是一个连接请求，或连接接受报文。SYN这个标志位只有在TCP建产连接时才会被置1，握手完成后SYN标志位被置0。

    终止FIN：用来释放一个连接。FIN=1表示：此报文段的发送方的数据已经发送完毕，并要求释放运输连接

    PS：ACK、SYN和FIN这些大写的单词表示标志位，其值要么是1，要么是0；ack、seq小写的单词表示序号。

|  |  |
| --- | --- |
| 字段 | 含义 |
| URG | 紧急指针是否有效。为1，表示某一位需要被优先处理 |
| ACK | 确认号是否有效，一般置为1。 |
| PSH | 提示接收端应用程序立即从TCP缓冲区把数据读走。 |
| RST | 对方要求重新建立连接，复位。 |
| SYN | 请求建立连接，并在其序列号的字段进行序列号的初始值设定。建立连接，设置为1 |
| FIN | 希望断开连接。 |

三次握手过程理解

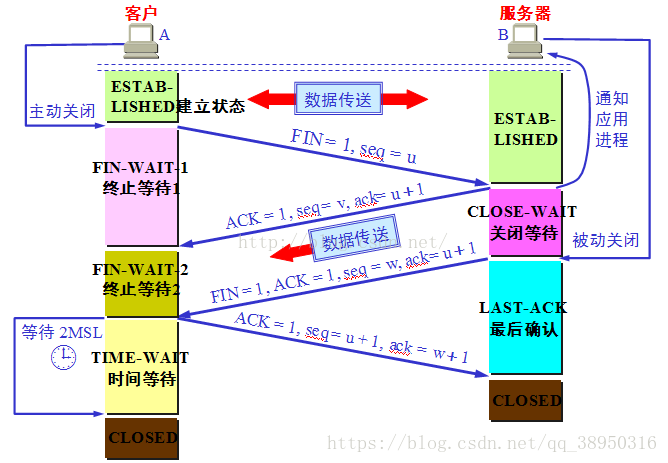


第一次握手：建立连接时，客户端发送syn包（syn=j）到服务器，并进入SYN\_SENT状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到syn包，必须确认客户的SYN（ack=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入SYN\_RECV状态；

第三次握手：客户端收到服务器的SYN+ACK包，向服务器发送确认包ACK(ack=k+1），此包发送完毕，客户端和服务器进入ESTABLISHED（TCP连接成功）状态，完成三次握手。

四次挥手过程理解



1）客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为seq=u（等于前面已经传送过来的数据的最后一个字节的序号加1），此时，客户端进入FIN-WAIT-1（终止等待1）状态。 TCP规定，FIN报文段即使不携带数据，也要消耗一个序号。

2）服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号seq=v，此时，服务端就进入了CLOSE-WAIT（关闭等待）状态。TCP服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。

3）客户端收到服务器的确认请求后，此时，客户端就进入FIN-WAIT-2（终止等待2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。

4）服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为seq=w，此时，服务器就进入了LAST-ACK（最后确认）状态，等待客户端的确认。

5）客户端收到服务器的连接释放报文后，必须发出确认，ACK=1，ack=w+1，而自己的序列号是seq=u+1，此时，客户端就进入了TIME-WAIT（时间等待）状态。注意此时TCP连接还没有释放，必须经过2∗∗MSL（最长报文段寿命）的时间后，当客户端撤销相应的TCB后，才进入CLOSED状态。

6）服务器只要收到了客户端发出的确认，立即进入CLOSED状态。同样，撤销TCB后，就结束了这次的TCP连接。可以看到，服务器结束TCP连接的时间要比客户端早一些。

 常见面试题

【问题1】为什么连接的时候是三次握手，关闭的时候却是四次握手？

答：因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

【问题2】为什么TIME\_WAIT状态需要经过2MSL(最大报文段生存时间)才能返回到CLOSE状态？

答：虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假象网络是不可靠的，有可以最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。在Client发送出最后的ACK回复，但该ACK可能丢失。Server如果没有收到ACK，将不断重复发送FIN片段。所以Client不能立即关闭，它必须确认Server接收到了该ACK。Client会在发送出ACK之后进入到TIME\_WAIT状态。Client会设置一个计时器，等待2MSL的时间。如果在该时间内再次收到FIN，那么Client会重发ACK并再次等待2MSL。所谓的2MSL是两倍的MSL(Maximum Segment Lifetime)。MSL指一个片段在网络中最大的存活时间，2MSL就是一个发送和一个回复所需的最大时间。如果直到2MSL，Client都没有再次收到FIN，那么Client推断ACK已经被成功接收，则结束TCP连接。

【问题3】为什么不能用两次握手进行连接？

答：3次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。

       现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机S和C之间的通信，假定C给S发送一个连接请求分组，S收到了这个分组，并发 送了确认应答分组。按照两次握手的协定，S认为连接已经成功地建立了，可以开始发送数据分组。可是，C在S的应答分组在传输中被丢失的情况下，将不知道S 是否已准备好，不知道S建立什么样的序列号，C甚至怀疑S是否收到自己的连接请求分组。在这种情况下，C认为连接还未建立成功，将忽略S发来的任何数据分 组，只等待连接确认应答分组。而S在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

【问题4】如果已经建立了连接，但是客户端突然出现故障了怎么办？

TCP还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为2小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔75秒钟发送一次。若一连发送10个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

## OSI参考模型

        今天我们先学习一下以太网最基本也是重要的知识——OSI参考模型。

 1、OSI的来源

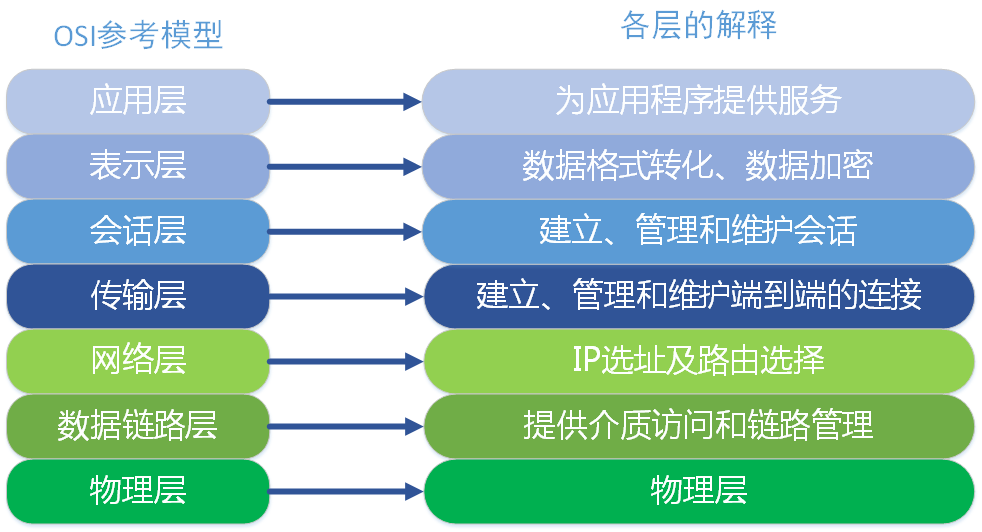
        OSI（Open System Interconnect），即开放式系统互联。 一般都叫OSI参考模型，是ISO（国际标准化组织）组织在1985年研究的网络互连模型。

        ISO为了更好的使网络应用更为普及，推出了OSI参考模型。其含义就是推荐所有公司使用这个规范来控制网络。这样所有公司都有相同的规范，就能互联了。

  2、OSI七层模型的划分

       OSI定义了网络互连的七层框架（物理层、数据链路层、网络层、传输层、会话层、表示层、应用层），即ISO开放互连系统参考模型。如下图。

        每一层实现各自的功能和协议，并完成与相邻层的接口通信。OSI的服务定义详细说明了各层所提供的服务。某一层的服务就是该层及其下各层的一种能力，它通过接口提供给更高一层。各层所提供的服务与这些服务是怎么实现的无关。



 3、各层功能定义

        这里我们只对OSI各层进行功能上的大概阐述，不详细深究，因为每一层实际都是一个复杂的层。后面我也会根据个人方向展开部分层的深入学习。这里我们就大概了解一下。我们从最顶层——应用层 开始介绍。整个过程以公司A和公司B的一次商业报价单发送为例子进行讲解。

<1>    应用层

        OSI参考模型中最靠近用户的一层，是为计算机用户提供应用接口，也为用户直接提供各种网络服务。我们常见应用层的网络服务协议有：HTTP，HTTPS，FTP，POP3、SMTP等。

        实际公司A的老板就是我们所述的用户，而他要发送的商业报价单，就是应用层提供的一种网络服务，当然，老板也可以选择其他服务，比如说，发一份商业合同，发一份询价单，等等。

<2>    表示层

        表示层提供各种用于应用层数据的编码和转换功能,确保一个系统的应用层发送的数据能被另一个系统的应用层识别。如果必要，该层可提供一种标准表示形式，用于将计算机内部的多种数据格式转换成通信中采用的标准表示形式。数据压缩和加密也是表示层可提供的转换功能之一。

        由于公司A和公司B是不同国家的公司，他们之间的商定统一用英语作为交流的语言，所以此时表示层（公司的文秘），就是将应用层的传递信息转翻译成英语。同时为了防止别的公司看到，公司A的人也会对这份报价单做一些加密的处理。这就是表示的作用，将应用层的数据转换翻译等。

<3>    会话层

        会话层就是负责建立、管理和终止表示层实体之间的通信会话。该层的通信由不同设备中的应用程序之间的服务请求和响应组成。

        会话层的同事拿到表示层的同事转换后资料，（会话层的同事类似公司的外联部），会话层的同事那里可能会掌握本公司与其他好多公司的联系方式，这里公司就是实际传递过程中的实体。他们要管理本公司与外界好多公司的联系会话。当接收到表示层的数据后，会话层将会建立并记录本次会话，他首先要找到公司B的地址信息，然后将整份资料放进信封，并写上地址和联系方式。准备将资料寄出。等到确定公司B接收到此份报价单后，此次会话就算结束了，外联部的同事就会终止此次会话。

<4>   传输层

        传输层建立了主机端到端的链接，传输层的作用是为上层协议提供端到端的可靠和透明的数据传输服务，包括处理差错控制和流量控制等问题。该层向高层屏蔽了下层数据通信的细节，使高层用户看到的只是在两个传输实体间的一条主机到主机的、可由用户控制和设定的、可靠的数据通路。我们通常说的，TCP UDP就是在这一层。端口号既是这里的“端”。

        传输层就相当于公司中的负责快递邮件收发的人，公司自己的投递员，他们负责将上一层的要寄出的资料投递到快递公司或邮局。

<5>   网络层

       本层通过IP寻址来建立两个节点之间的连接，为源端的运输层送来的分组，选择合适的路由和交换节点，正确无误地按照地址传送给目的端的运输层。就是通常说的IP层。这一层就是我们经常说的IP协议层。IP协议是Internet的基础。

        网络层就相当于快递公司庞大的快递网络，全国不同的集散中心，比如说，从深圳发往北京的顺丰快递（陆运为例啊，空运好像直接就飞到北京了），首先要到顺丰的深圳集散中心，从深圳集散中心再送到武汉集散中心，从武汉集散中心再寄到北京顺义集散中心。这个每个集散中心，就相当于网络中的一个IP节点。

<6>   数据链路层

        将比特组合成字节,再将字节组合成帧,使用链路层地址 (以太网使用MAC地址)来访问介质,并进行差错检测。

     数据链路层又分为2个子层：逻辑链路控制子层（LLC）和媒体访问控制子层（MAC）。

        MAC子层处理CSMA/CD算法、数据出错校验、成帧等；LLC子层定义了一些字段使上次协议能共享数据链路层。 在实际使用中，LLC子层并非必需的。

        这个没找到合适的例子

<7>  物理层

        实际最终信号的传输是通过物理层实现的。通过物理介质传输比特流。规定了电平、速度和电缆针脚。常用设备有（各种物理设备）集线器、中继器、调制解调器、网线、双绞线、同轴电缆。这些都是物理层的传输介质。

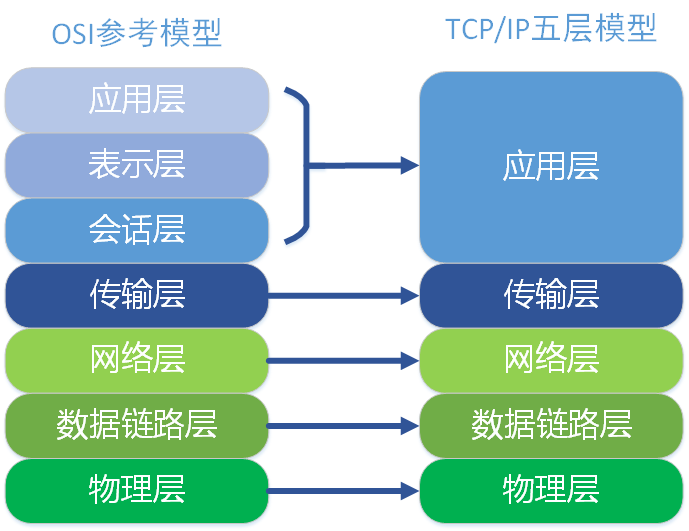
         快递寄送过程中的交通工具，就相当于我们的物理层，例如汽车，火车，飞机，船。

4、通信特点：对等通信

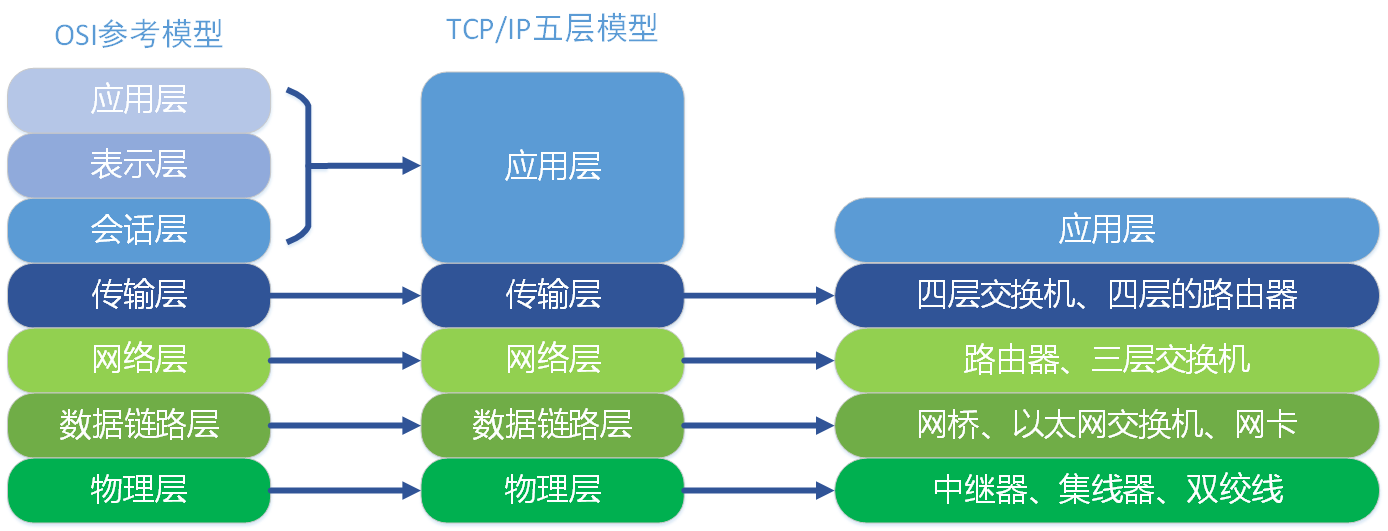
对等通信，为了使数据分组从源传送到目的地，源端OSI模型的每一层都必须与目的端的对等层进行通信，这种通信方式称为对等层通信。在每一层通信过程中，使用本层自己协议进行通信。

## TCP/IP五层模型

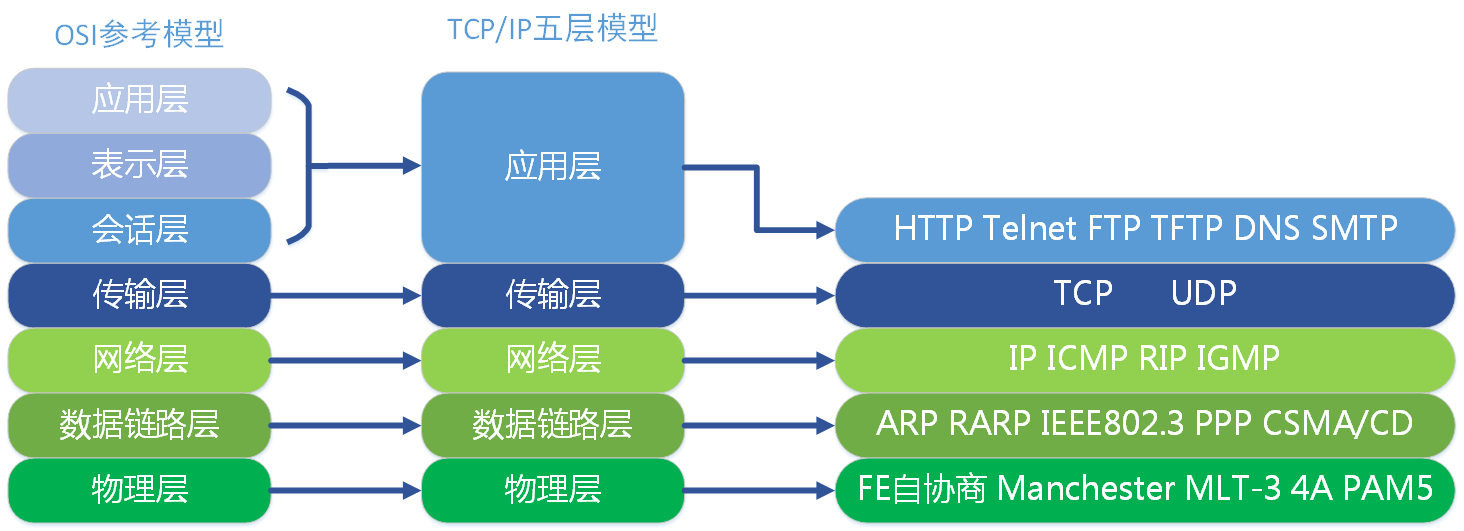
    TCP/IP五层协议和OSI的七层协议对应关系如下。



    在每一层都工作着不同的设备，比如我们常用的交换机就工作在数据链路层的，一般的路由器是工作在网络层的。



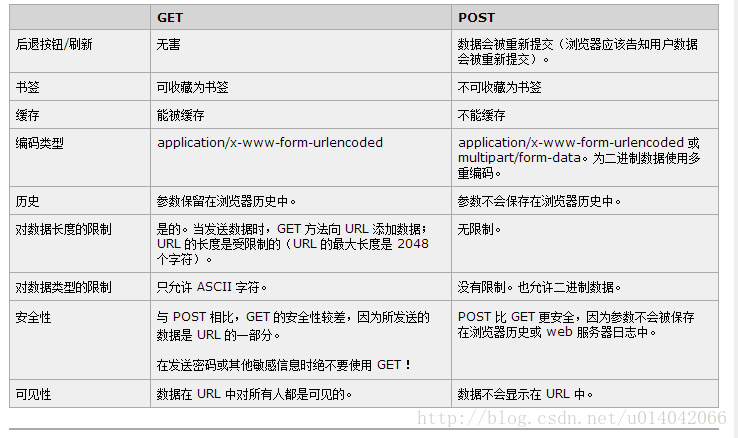
在每一层实现的协议也各不同，即每一层的服务也不同.下图列出了每层主要的协议。其中每层中具体的协议，我会在后面的逐一学习。



**HTTP 响应码，比如 200, 302, 404。**

1xx：信息，请求收到，继续处理  
2xx：成功，行为被成功地接受、理解和采纳  
3xx：重定向，为了完成请求，必须进一步执行的动作  
4xx：客户端错误，请求包含语法错误或者请求无法实现  
5xx：服务器错误，服务器不能实现一种明显无效的请求  
200 ok 一切正常  
302 Moved Temporatily 文件临时移出  
404 not found

**简述 Http 请求 get 和 post 的区别以及数据包格式**



1. http是无状态通信，http的请求方式有哪些，可以自己定义新的请求方式么。

HTTP是无状态的，它的底层协议是由状态的TCP，但是HTTP的一次完整协议动作，里面是使用有状态的TCP协议来完成的。而每次协议动作之间没有任何关系。例如：第7次请求HTTP协议包，并不知道，这个包是为了什么？它或许是因为上次没有请求成功而重传，或许是上次的后续请求，或许是其他的，这些HTTP自身都不知道。

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。



http中，get post的区别

GET在浏览器回退时是无害的，而POST会再次提交请求。

GET产生的URL地址可以被Bookmark，而POST不可以。

GET请求会被浏览器主动cache，而POST不会，除非手动设置。

GET请求只能进行url编码，而POST支持多种编码方式。

GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。

GET请求在URL中传送的参数是有长度限制的，而POST么有。

对参数的数据类型，GET只接受ASCII字符，而POST没有限制。

GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。

GET参数通过URL传递，POST放在Request body中。

GET和POST本质上没有区别

GET和POST是什么？HTTP协议中的两种发送请求的方法。

HTTP是什么？HTTP是基于TCP/IP的关于数据如何在万维网中如何通信的协议。

HTTP的底层是TCP/IP。所以GET和POST的底层也是TCP/IP，也就是说，GET/POST都是TCP链接。GET和POST能做的事情是一样一样的。你要给GET加上request body，给POST带上url参数，技术上是完全行的通的。

因此，GET和POST本质上就是TCP链接，并无差别。但是由于HTTP的规定和浏览器/服务器的限制，导致他们在应用过程中体现出一些不同。

GET和POST还有一个重大区别，简单的说：

GET产生一个TCP数据包；POST产生两个TCP数据包。

对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

因为POST需要两步，时间上消耗的要多一点，看起来GET比POST更有效。因此Yahoo团队有推荐用GET替换POST来优化网站性能。但这是一个坑！跳入需谨慎。为什么？

1. GET与POST都有自己的语义，不能随便混用。

2. 据研究，在网络环境好的情况下，发一次包的时间和发两次包的时间差别基本可以无视。而在网络环境差的情况下，两次包的TCP在验证数据包完整性上，有非常大的优点。

3. 并不是所有浏览器都会在POST中发送两次包，Firefox就只发送一次。

8. 说说http,tcp,udp之间关系和区别。

TCP/IP是个协议组，可分为四个层次：网络接口层、网络层、传输层和应用层。

在网络层有IP协议、ICMP协议、ARP协议、RARP协议和BOOTP协议。

在传输层中有TCP协议与UDP协议。

在应用层有HTTP、FTP、TELNET、SMTP、DNS等协议。

TCP  传送控制协议(Transmission Control Protocol)：

TCP是传输层的一个协议，基于IP协议，用来传输类似HTTP的信息。如果把IP协议类比为一个“公路”的话，那TCP协议可以看成是在公路上行驶的“卡车”。TCP协议是面向连接的协议，通过三次握手机制，尽量保证连接的可靠性。tcp的链接需要进行三次握手，释放连接需要四次挥手。

UDP 用户数据报协议 （User Datagram Protocol） ：

　　UDP也是传输层的一个协议。但是与TCP不同的是，UDP不是面向连接的，并不保证传输的可靠性，没有TCP的建立连接的三次握手机制，对于传输效率上面有了提升。

个人理解：

这个就比较简单粗暴了，A要给B传数据，然后就直接传了。

HTTP 超文本传输协议（HyperText Transfer Protocal）：

　　HTTP是在应用层的一个协议，本身就是一个协议，是从Web服务器传输超文本到本地浏览器的传输协议。

　　HTTP协议基于请求\响应模型的，并且是基于TCP协议的。

　　HTTP连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接。从建立连接到关闭连接的过程称为“一次连接”。

HTTP协议、  HTTPS协议，SSL协议及完整交互过程；

SSL

1.安全套接字（Secure Socket Layer，SSL）协议是Web浏览器与Web服务器之间安全交换信息的协议。

2.SSL协议的三个特性

Ø  保密：在握手协议中定义了会话密钥后，所有的消息都被加密。

Ø  鉴别：可选的客户端认证，和强制的服务器端认证。

Ø  完整性：传送的消息包括消息完整性检查（使用MAC）。

3.SSL的位置

HTTPS

1.HTTPS基于SSL的HTTP协议。

2.HTTPS使用与HTTP不同的端口(HTTP:80 ， HTTPS:443，一个加密、身份验证层（HTTP与TCP之间）)。

3.提供了身份验证与加密通信方法，被广泛用于互联网上安全敏感的通信。

交互过程

客户端在使用HTTPS方式与Web服务器通信时有以下几个步骤，如图所示。

1)    客户端请求建立SSL连接，并将自己支持的一套加密规则发送给网站。

2)    网站从中选出一组加密算法与HASH算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含了网站地址，加密公钥，以及证书的颁发机构等信息

3)    获得网站证书之后浏览器要做以下工作：

Ø  验证证书的合法性

Ø  如果证书受信任，浏览器会生成一串随机数的密码，并用证书中提供的公钥加密。

Ø  使用约定好的HASH计算握手消息，

Ø  使用生成的随机数对消息进行加密，最后将之前生成的所有信息发送给网站。

4)    网站接收浏览器发来的数据之后要做以下的操作：

Ø  使用自己的私钥将信息解密取出密码

Ø  使用密码解密浏览器发来的握手消息，并验证HASH是否与浏览器发来的一致。

Ø  使用密码加密一段握手消息，发送给浏览器

5)    浏览器解密并计算握手消息的HASH，如果与服务端发来的HASH一致，此时握手结束。

6)    使用随机密码和对称加密算法对传输的数据加密，传输。

4.     加密与HASH算法如下：

1)     非对称加密算法：RSA，DSA/DSS，用于在握手过程中加密生成的密码。

2)     对称加密算法：AES，RC4，3DES，用于对真正传输的数据进行加密。

3)     HASH算法：MD5，SHA1，SHA256，验证数据的完整性。

5.     HTTP与HTTPS的区别：

1)     https协议需要申请证书。

2)     http是超文本传输协议，明文传输；https使用的是具有安全性的SSL加密传输协议。

3)     http端口80,；https端口443。

4)     http连接简单无状态；https由SSL+HTTP协议构件的可进行加密传输、身份验证的网络协议。

TCP与UDP区别？

udp: a、是面向无连接, 将数据及源的封装成数据包中,不需要建立建立连接

b、每个数据报的大小在限制64k内

c、因无连接,是不可靠协议

d、不需要建立连接,速度快

tcp： a、建议连接，形成传输数据的通道.

b、在连接中进行大数据量传输，以字节流方式

c 通过三次握手完成连接,是可靠协议

d 必须建立连接m效率会稍低

聊天、网络视频会议、桌面共享用的就是 udp

Xml与JSON区别

数据交换格式

区别:

xml是重量级、json是轻量级

xml比较占带宽、json占带宽小,易于压缩

json在webservice 用的比较少、xml用的较多

相同：

两者都用在项目交互下 例如 移动app接口用的就是json、在web项目中与其他项目对接用xml较多。

json常用解析方法 gson、jsonobject、jackson等 xml dom sax pull 解析

TCP与UDP区别？

udp: a、是面向无连接, 将数据及源的封装成数据包中,不需要建立建立连接

b、每个数据报的大小在限制64k内

c、因无连接,是不可靠协议

d、不需要建立连接,速度快

tcp： a、建议连接，形成传输数据的通道.

b、在连接中进行大数据量传输，以字节流方式

c 通过三次握手完成连接,是可靠协议

d 必须建立连接m效率会稍低

聊天、网络视频会议、桌面共享用的就是 udp

说说三次握手?

1）第一次握手：建立连接时，客户端A发送SYN包（SYN=j）到服务器B，并进入SYN\_SEND状态，等待服务器B确认。

（2）第二次握手：服务器B收到SYN包，必须确认客户A的SYN（ACK=j+1），同时自己也发送一个SYN包（SYN=k），即SYN+ACK包，此时服务器B进入SYN\_RECV状态。

（3）第三次握手：客户端A收到服务器B的SYN＋ACK包，向服务器B发送确认包ACK（ACK=k+1），此包发送完毕，客户端A和服务器B进入ESTABLISHED状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据。

什么是Webserivce?

Webservice就是提供不同的平台相互通讯，基于Soap协议。

Web service 就是一个应用程序，它向外界暴露出一个能够通过Web进行调用的API。

SOAP是一种简单基于xml的轻量协议，用户web上交换结构化信息和类型信息。

soap请求是HTTP POST的一个专用版本，遵循一种特殊的xml消息格式Content-type设置为: text/xml任何数据都可以xml化。

WebService实现原理是？

HTTP协议+XML

说一下什么是Http协议？

对器客户端和 服务器端之间数据传输的格式规范，格式简称为“超文本传输协议”。

什么是Http协议无状态协议？怎么解决Http协议无状态协议？（曾经去某创业公司问到）

1、无状态协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息

2、无状态协议解决办法： 通过1、Cookie 2、通过Session会话保存。

说一下Http协议中302状态(阿里经常问)

http协议中，返回状态码302表示重定向。

这种情况下，服务器返回的头部信息中会包含一个 Location 字段，内容是重定向到的url

Http协议有什么组成？

请求报文包含三部分：

a、请求行：包含请求方法、URI、HTTP版本信息

b、请求首部字段

c、请求内容实体

响应报文包含三部分：

a、状态行：包含HTTP版本、状态码、状态码的原因短语

b、响应首部字段

c、响应内容实体

Http协议中有那些请求方式？

GET： 用于请求访问已经被URI（统一资源标识符）识别的资源，可以通过URL传参给服务器

POST：用于传输信息给服务器，主要功能与GET方法类似，但一般推荐使用POST方式。

PUT： 传输文件，报文主体中包含文件内容，保存到对应URI位置。

HEAD： 获得报文首部，与GET方法类似，只是不返回报文主体，一般用于验证URI是否有效。

DELETE：删除文件，与PUT方法相反，删除对应URI位置的文件。

OPTIONS：查询相应URI支持的HTTP方法。

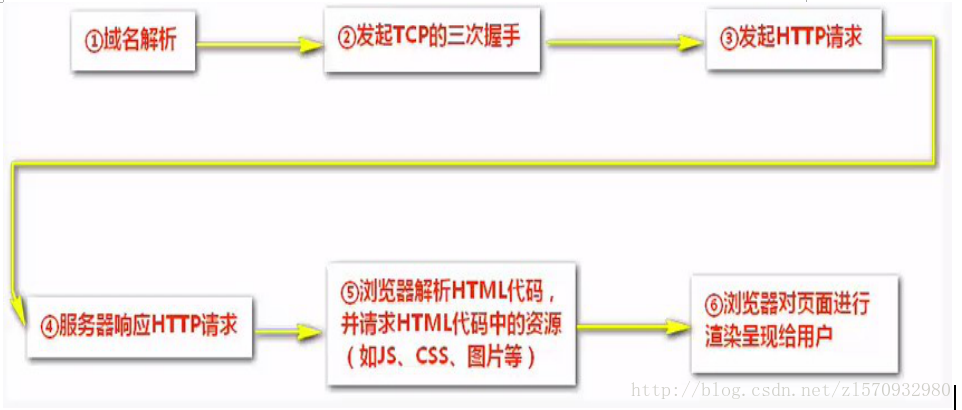
Http协议中Http1.0与1.1区别？

在http1.0中，当建立连接后，客户端发送一个请求，服务器端返回一个信息后就关闭连接，当浏览器下次请求的时候又要建立连接，显然这种不断建立连接的方式，会造成很多问题。

5.在http1.1中，引入了持续连接的概念，通过这种连接，浏览器可以建立一个连接之后，发送请求并得到返回信息，然后继续发送请求再次等到返回信息，也就是说客户端可以连续发送多个请求，而不用等待每一个响应的到来。

Http协议实现原理机制？

2.1、整个流程步骤：



2.2、域名解析过程



2.3、三次握手过程

     2.4、发起HTTP请求

     2.5、响应HTTP请求并得到HTML代码

     2.6、浏览器解析HTML代码

     2.7、浏览器对页面进行渲染呈现给用户

get与post请求区别？（初级程序员必备问题）

区别一：

  get重点在从服务器上获取资源，post重点在向服务器发送数据；

区别二：

   get传输数据是通过URL请求，以field（字段）= value的形式，置于URL后，并用”?”连接，多个请求数据间用”&”连接，如http://127.0.0.1/Test/login.action?name=admin&password=admin，这个过程用户是可见的；

post传输数据通过Http的post机制，将字段与对应值封存在请求实体中发送给服务器，这个过程对用户是不可见的；

区别三：

   Get传输的数据量小，因为受URL长度限制，但效率较高；

Post可以传输大量数据，所以上传文件时只能用Post方式；

区别四：

   get是不安全的，因为URL是可见的，可能会泄露私密信息，如密码等；

post较get安全性较高；

区别五：

   get方式只能支持ASCII字符，向服务器传的中文字符可能会乱码。

post支持标准字符集，可以正确传递中文字符。

9、Http请求报文与响应报文格式？

请求报文包含三部分：

a、请求行：包含请求方法、URI、HTTP版本信息

b、请求首部字段

c、请求内容实体

响应报文包含三部分：

a、状态行：包含HTTP版本、状态码、状态码的原因短语

b、响应首部字段

c、响应内容实体

10、常见Http协议状态？

200：请求被正常处理

204：请求被受理但没有资源可以返回

206：客户端只是请求资源的一部分，服务器只对请求的部分资源执行GET方法，相应报文中通过Content-Range指定范围的资源。

301：永久性重定向

302：临时重定向

303：与302状态码有相似功能，只是它希望客户端在请求一个URI的时候，能通过GET方法重定向到另一个URI上

304：发送附带条件的请求时，条件不满足时返回，与重定向无关

307：临时重定向，与302类似，只是强制要求使用POST方法

400：请求报文语法有误，服务器无法识别

401：请求需要认证

403：请求的对应资源禁止被访问

404：服务器无法找到对应资源

500：服务器内部错误

503：服务器正忙

Http协议首部字段？

a、通用首部字段（请求报文与响应报文都会使用的首部字段）

Date：创建报文时间

Connection：连接的管理

Cache-Control：缓存的控制

Transfer-Encoding：报文主体的传输编码方式

b、请求首部字段（请求报文会使用的首部字段）

Host：请求资源所在服务器

Accept：可处理的媒体类型

Accept-Charset：可接收的字符集

Accept-Encoding：可接受的内容编码

Accept-Language：可接受的自然语言

c、响应首部字段（响应报文会使用的首部字段）

Accept-Ranges：可接受的字节范围

Location：令客户端重新定向到的URI

Server：HTTP服务器的安装信息

d、实体首部字段（请求报文与响应报文的的实体部分使用的首部字段）

Allow：资源可支持的HTTP方法

Content-Type：实体主类的类型

Content-Encoding：实体主体适用的编码方式

Content-Language：实体主体的自然语言

Content-Length：实体主体的的字节数

Content-Range：实体主体的位置范围，一般用于发出部分请求时使用

Http与Https优缺点？

a、通信使用明文不加密，内容可能被窃听，也就是被抓包分析。

b、不验证通信方身份，可能遭到伪装

c、无法验证报文完整性，可能被篡改

HTTPS就是HTTP加上加密处理（一般是SSL安全通信线路）+认证+完整性保护

Http优化

利用负载均衡优化和加速HTTP应用

利用HTTP Cache来优化网站

Http协议有那些特征？

1、支持客户/服务器模式；2、简单快速；3、灵活；4、无连接；5、无状态；