一、计算机网络体系结构

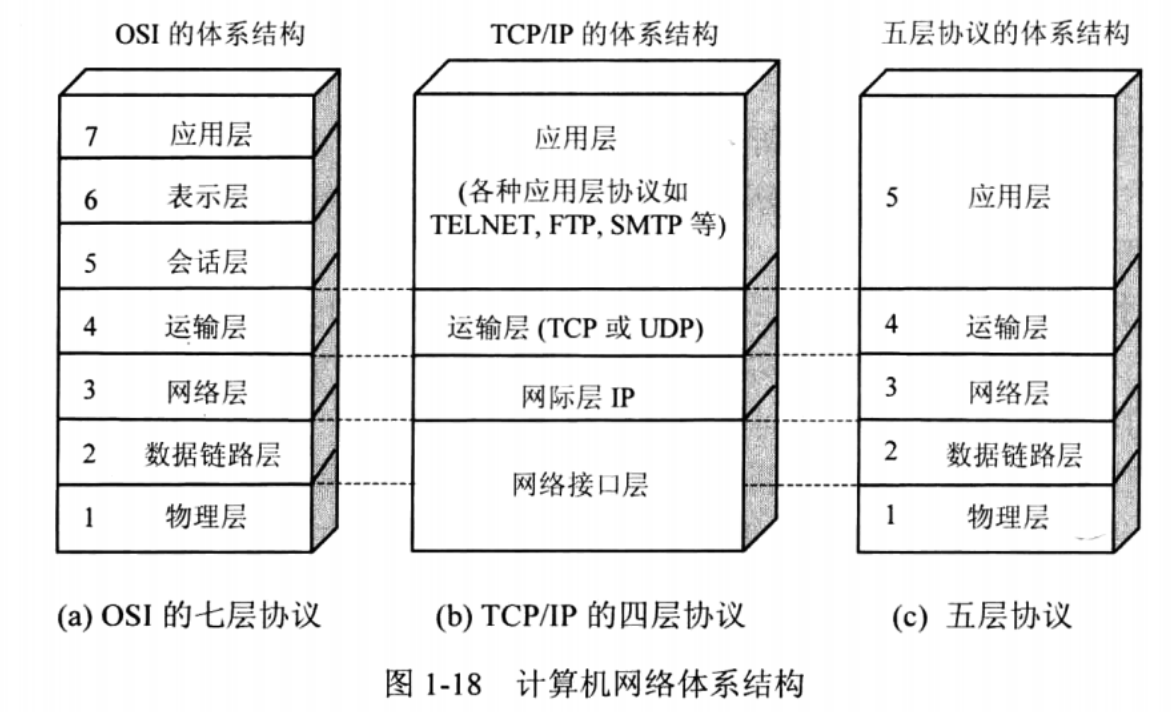


图 1-18　计算机网络体系结构

（1）应用层

应用层的任务是通过应用进程间的交互来完成特定网络应用。应用层协议定义的是应用进程间通信和交互的规则。这里的进程就是指主机中正在运行的程序。如：支持万维网的HTTP协议，支持电子邮件的SMTP协议，支持文件传送的FTP协议，等等。我们将应用层交互的数据单元称为报文（message）。

（2）传输层

传输层的任务就是负责向两个主机中进程之间的通信提供通用的数据传输服务。应用进程利用该服务传送应用层报文。所谓通用，是指并不针对某个特定网络应用，而是多种应用可以使用同一个传输层服务。运输层主要有两种协议：

- 传输控制协议TCP（Transmission Control Protocol）：提供面向连接的、可靠的数据传输服务，其数据传输的单位是报文段（segment）
- 用户数据报协议UDP（User Datagram Protocol）：提供无连接的、尽最大努力的数据传输服务（不保证数据传输的可靠性），其数据传输的单位是用户数据报。

（3）网络层

网络层负责为分组交换网上的不同主机提供通信服务。在发送数据时，网络层把运输层产生的报文段或用户数据报封装成分组或包（packet）进行传送。在TCP/IP体系中，由于网络层使用IP协议，因此分组也叫作IP数据报，或简称为数据报。

（4）数据链路层

（5）物理层

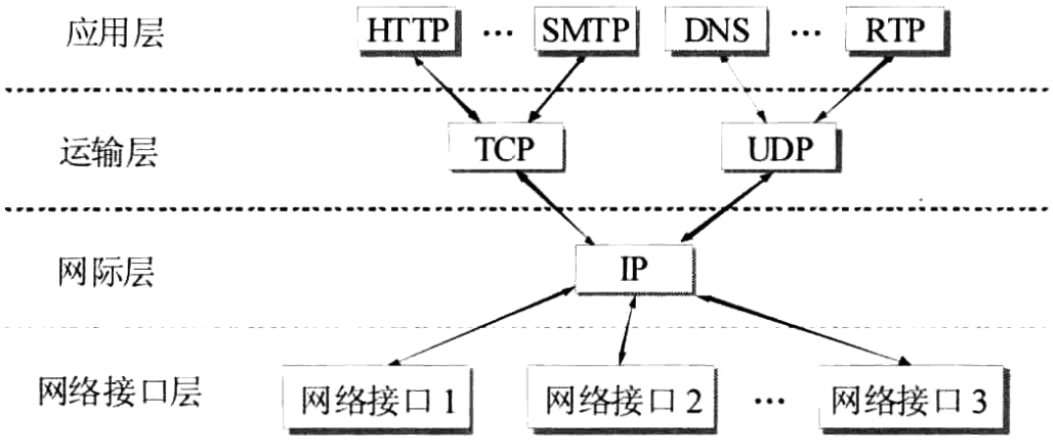


图 1-24　沙漏计时器形状的 TCP/IP 协议族示意

二、网络层

因特网采用的设计思路是：网络层向上只提供简单灵活的、无连接的、尽最大努力交付的数据报服务。网络在发送分组。网络在发送分组时不需要先建立连接。每一个分组（也就是IP数据报）独立发送，与其前后的分组无关（不进行编号）。网络层不提供服务质量的承诺。所发送的分组可能出错、丢失、重复和失序（即不按时序到达终点），当然也不保证分组交付的时限。

整个因特网就是一个单一、抽象的网络。IP地址就是给因特网上的每一个主机（或路由器）的每一个接口分配一个在全世界范围是唯一的32位标识符。IP地址现在由因特网名字和数字分配机构ICANN进行分配。

IP地址的编址方法共经过了三个历史阶段：

- 分类的IP地址，这是最基本的编址方法
- 子网划分，这是对基本的编址方法的改进
- 构成超网，这是比较新的五分类编址方法

1、分类的IP地址

所谓的“分类IP地址”就是将IP地址划分为若干个固定类，每一类地址都由两个固定长度的字段组成，其中第一个字段是网络号（net-id），它标志主机（或路由器）所连接到的网络。一个网络号在整个因特网范围内必须是唯一的。第二个字段是主机号（host-id），它标志该主机（或路由器）。一个主机号在它前面的网络号所指明的网络范围内必须是唯一的。

$$IP_{\text{地址}} ::= \{< \text{网络号} >, < \text{主机号} >\}$$

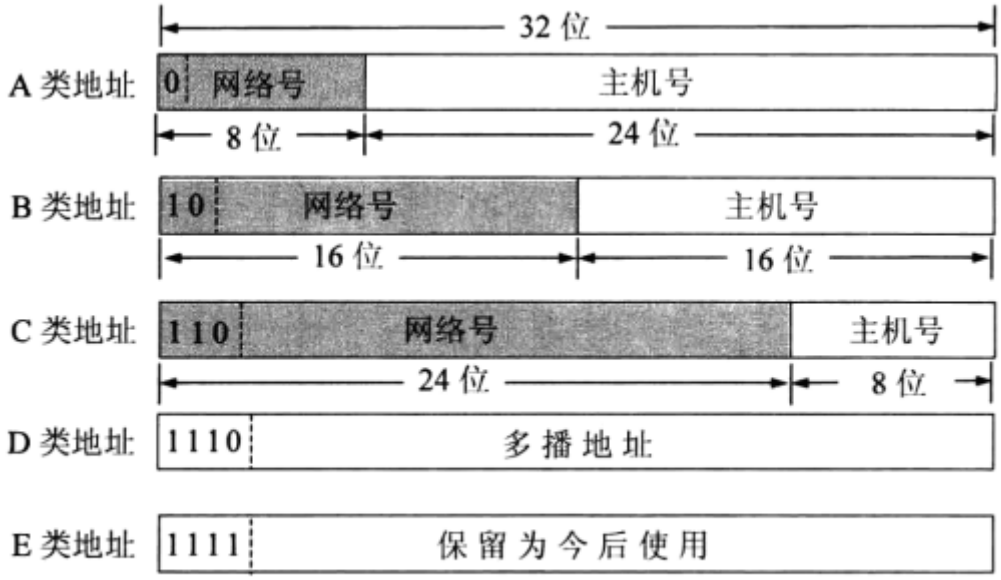


图 4-5　IP 地址中的网络号字段和主机号字段

上图中A、B、C类地址都是单播地址（一对一通信）。

- A类、B类和C类地址的网络号字段（图中灰色部分）分别为1,2,3字节长，而在网络号字段的最前面有1~3位的类别位，其数值分别规定为0,10和110。
- A类、B类和C类地址的主机号字段分别为3个、2个和1个字节长。
- D类地址（前4位是1110）用于多播（一对多通信）。
- E类地址（前4位是1111）保留为以后用

对于A类地址，只有7位可供使用，但可指派的网络号是126个（2^7-2），减2的原因是因为：第一，IP地址的全0表示“这个（this）”。网络号字段全0的IP地址是个保留地址，意思是“本网络”；第二，网络号为127（即01111111）保留作为本地软件环回测试本机主机进程之间通信之用。A类地址的主机号占3字节，每一个A类网络中的最大主机数是$2^{24}-2$，即16777214，减2的原因是：全0的主机号字段表示该IP地址是“本主机”所连接到的单个网络地址（例如，一主机的IP地址为5.6.7.8，则该主机所在的网络地址就是5.0.0.0），而全1表示“所有的（all）”，因此全1的主机号字段表示该网络上的所有主机。

对于B类地址的网络号字段有2个字节，但前面两位（10）已经固定了，只剩下14位可以进行分配。因为网络号字段前面的14位不论怎样取值也不可能出现使整个2字节的网络号字段称为全0或全1，因此不存在网络总数减2的问题。但实际上B类网络地址128.0.0.0是不指派的，而可以指派的B类最小网络地址是128.1.0.0。因此B类地址可以指派的网络数为$2^{14}-1$，即16383。B类地址的最大主机数是$2^{16}-2$，即65534。这里需要减2是因为要扣除全0和全1的主机号。

对于C类地址，有3个字节的网络号字段，最前面的3位是（110），还有21位可以进行分类。C类网络地址192.0.0.0也是不指派的，可以指派的C类最小网络地址是192.0.1.0，因此C类地址可指派的网络总数是$2^{21}-1$，即2097151。每个C类地址的最大主机数是2^8-2，即254。整个C类地址空间共约2^{29}个地址。

2、划分子网

从1985年起在IP地址中增加了一个“子网号字段”，使得两级IP地址变为三级IP地址，这种做法叫做划分子网或子网寻址或子网路由选择。

划分子网的基本思路是：

（1）一个拥有许多物理网络的单位，可将所属的物理网络划分为若干个子网（subnet）。划分子网纯属一个单位内部的事情。本单位以外的网络看不见这个网络是由多少个子网组成，因为这个单位对外仍然表现为一个网络。

（2）划分子网的方法是从网络的主机号借用若干位作为子网号subnet-id，当然主机号也就相应减少了同样的位数。于是两级IP地址在本单位内部就变为三级IP地址：网络号、子网号和主机号，可以用以下计法表示：

$$IP\text{地址} ::= \{< \text{网络号} >, < \text{子网号} >, < \text{主机号} >\}$$

（3）凡是从其他网络发送给本单位某个主机的IP数据报，仍然是根据IP数据报的目的网络号连接在本单位网络的路由器。但此路由器收到IP数据报后，再按目的的网络号和子网号找到目的子网，把数据报交付目的主机。

IP数据报的首部无法看出源主机或目的主机所连接的网络是否进行了子网划分，在确定子网的时候使用子网掩码进行确定。把子网掩码和IP地址进行逐位的“与”运算（AND），就立即得出网络地址来。这样路由器处理到来的分组时就可采用同样的算法。

A类地址的默认子网掩码是255.0.0.0，或0xFF000000；

B类地址的默认子网掩码是255.255.0.0，或0xFFFF0000；

C类地址的默认子网掩码是255.255.255.0，或FFFFFF00

3、无分类编址CIDR（构成超网）

三、运输层

网络层是为主机之间提供逻辑通信，而运输层为应用进程之间提供端到端的逻辑通信。运输层向高层用户屏蔽了下面网络核心的细节，它使应用进程看见的就是好像在两个运输层实体之间有一条端到端的逻辑通信信道。当运输层采用面向连接的TCP协议时，尽管下面的网络是不可靠的，但这种逻辑通信信道就相当于一条全双工的可靠信道。但当运输层无连接的UDP协议时，这种逻辑通信信道仍然是一条不可靠信道。

在TCP/IP体系中，根据使用的协议是TCP或UDP，分别称之为TCP报文段（segment）或UDP用户数据报。

UDP在传送数据之前不需要先建立连接。远地主机的运输层在收到UDP报文后，不需要给出任何确认。虽然UDP不提供可靠交付，但某些情况下UDP却是一种最有效的工作方式。

TCP则是面向连接的服务。在传送数据之前必须先建立连接，数据传送结束后要释放连接。TCP不提供广播或多播服务。由于TCP要提供可靠的、面向连接的运输服务，因此不可避免地增加许多的开销。

表 5-1　使用 UDP 和 TCP 协议的各种应用和应用层协议

应　　用	应用层协议	运输层协议
名字转换	DNS（域名系统）	UDP
文件传送	TFTP（简单文件传送协议）	UDP
路由选择协议	RIP（路由信息协议）	UDP
IP 地址配置	DHCP（动态主机配置协议）	UDP
网络管理	SNMP（简单网络管理协议）	UDP
远程文件服务器	NFS（网络文件系统）	UDP
IP 电话	专用协议	UDP
流式多媒体通信	专用协议	UDP
多播	IGMP（网际组管理协议）	UDP
电子邮件	SMTP（简单邮件传送协议）	TCP
远程终端接入	TELNET（远程终端协议）	TCP
万维网	HTTP（超文本传送协议）	TCP
文件传送	FTP（文件传送协议）	TCP

应用层所有的应用进程都可以通过运输层再传送到IP层（网络层），这就是复用。运输层从IP层收到数据后必须交付指明的应用进程，这就是分用。

解决分用问题的方法是在运输层使用协议端口号（protocol port number），或通常简称为端口（port）。这就是说，虽然通信的终点是应用进程，但我们只要把传送的报文交到目的主机的某一合适的目的端口，剩下的工作（即最后交付目的进程）就由TCP来完成。

请注意，这种在协议栈层间的协议端口是软件端口，和路由器或交换机上的硬件端口是完全不同的概念。硬件端口是不同设备进行交互的端口，而软件端口是应用层各种协议进程与运输实体进行层间交互的一种地址。在TCP和UDP首部的格式中，都有源端口和目的端口这两个重要字段。

TCP/IP的运输层用一个16位端口号来标志一个端口。可允许有65535个不同的端口号。运输层的端口号共分为两大类：

（1）服务器端使用的端口号：这里又分为两大类，最重要的一类叫做熟知端口号或系统端口号，数值是0~1023，这些数值可以在www.iana.org查到。

表 5-2　常用的熟知端口号

应用程序	FTP	TELNET	SMTP	DNS	TFTP	HTTP	SNMP	SNMP (trap)
熟知端口号	21	23	25	53	69	80	161	162

另一类是登记端口号，数值为1024~49151。这类端口号是为没有熟知的应用程序使用的。

（2）客户端使用的端口号：

数值为49152~65535。由于这类端口号仅在客户进程运行时才动态选择，因此也叫做短暂端口号。这类端口号是留给客户进程选择暂时使用。

1.用户数据报协议UDP

UDP的特点是：

- **UDP是无连接的**
- **UDP使用尽最大努力交付**。即不保证可靠交付，因此主机不需要维持复杂的连接状态表
- **UDP是面向报文的**
- **UDP没有拥塞控制**，因此网络出现拥塞不会使源主机的发送速率降低。这对某些实时应用是很重要的。很多的实施应用（如：IP电话、**实时视频会议**等）要求源主机以恒定的速率发送数据，并且允许在网络发生拥塞时丢失一些数据，但不允许数据有太大的时延。
- **UDP支持一对一、一对多、多对一和多对多的交互通信。**
- **UDP的头部开销小，只有8个字节，比TCP的20个字节的首部要短**。虽然某些实时应用需要使用没有拥塞控制的UDP，但当很多的源主机同时都向网络发送高速率的实时视频流时，网络就有可能发生拥塞，结果大家都无法正常接收。因此，不使用拥塞控制功能的UDP有可能会引起网络产生严重的拥塞问题。还有一些使用UDP的实时应用，需要对UDP的不可靠的传输进行适当的改进，以减少数据的丢失。这种情况下，应用进程本身可以不影响应用的实时性的前提下，增加一些提高可靠性的措施，如采用前向纠错或重传丢失的报文。

用户数据报UDP有两个字段：数据字段和首部字段。首部字段很简单，只有8个字节，由四个字段组成，每个字段的长度都是两个字节。

- 源端口：在需要对方回信时选用。不需要时可用全0
- 目的端口：这在终点交付报文时必须要使用到
- 长度：UDP用户数据报的长度，其最小值是8（仅有首部）
- 检验和：检测UDP用户数据报在传输中是否有错，有错就丢弃

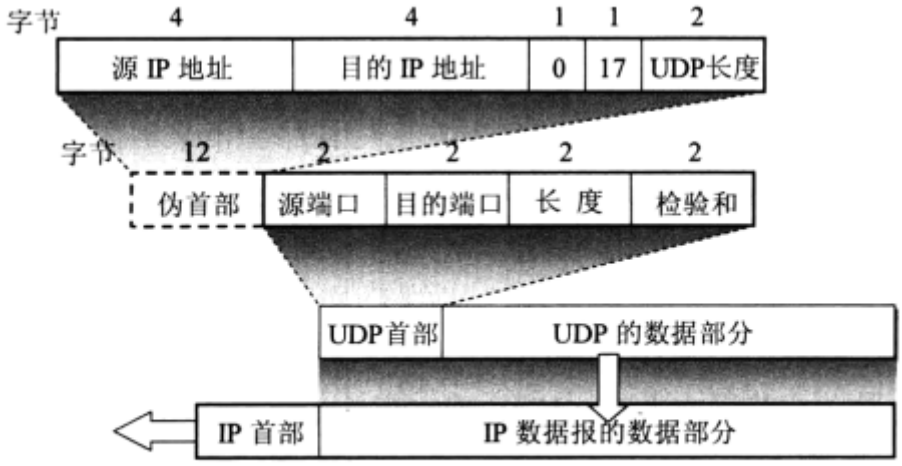


图 5-5　UDP 用户数据报的首部和伪首部

2.传输控制协议TCP

TCP的特点是：

- **TCP是面向连接的运输层协议**。应用在使用TCP协议之前，必须建立TCP连接。在传送数据完毕后，必须释放已经建立的TCP连接。
- **每一条TCP连接只能有两个端点（endpoint），每一条TCP连接只能是点对点的**（一对一）。
- **TCP提供可靠交付的服务**。通过TCP连接传送的数据，无差错、不丢失、不重复、并且按序到达

- **TCP提供全双工通信**。TCP允许通信双方的应用进程在任何时候都能发送数据。TCP连接的两端都设有发送缓存和接受缓存，用来临时存放双向通信的数据。在发送时，应用程序会吧数据传送给TCP缓存后，就可以做自己的事，而TCP在合适的时候把数据发送出去。在接收时，TCP把收到的数据放入缓存，上层应用进程在合适的时候读取缓存中的数据。
- **面向字节流**。TCP中的流（stream）指的是流入到进程或从进程流出的字节序列。

TCP把连接作为最基本的抽象。TCP连接的端点叫做套接字（socket）或插口。根据RFC 793的定义：端口号拼接到IP地址即构成了套接字。因此，套接字的表示方法是在点分十进制的IP地址后面写上端口号，中间用冒号或逗号隔开。

$$套接字 socket = (IP地址：端口号)$$

每一条TCP连接唯一地被通信两端的两个端点（即两个套接字）所确定。即：

$$TCP连接 ::= \{socket_1, socket_2\} = \{(IP_1, port_1), (IP_2, port_2)\}$$

TCP连接就是由协议软件所提供的一种抽象。TCP连接的端点是个很抽象的套接字，即（IP地址，端口号）。同一个IP地址可以有多个不同的TCP连接，而同一个端口也可以出现在多个不同的TCP连接中。

（1）可靠传输的工作原理

1）停止等待协议

TCP发送的报文段是交给IP层传送的，但IP层只能提供尽最大努力服务，也就是说，TCP下面的网络所提供的是不可靠的传输。因此，TCP必须采用适当的措施才能使得两个运输层之间的通信变得可靠。

这里探讨可靠传输的原理，因此把每次传送的数据单元都称为分组，并不考虑数据是在哪一个层次上传送的。“停止等待”就是每发送完一次分组就停止发送，等待对方的确认。在收到确认后再发送下一个分组。

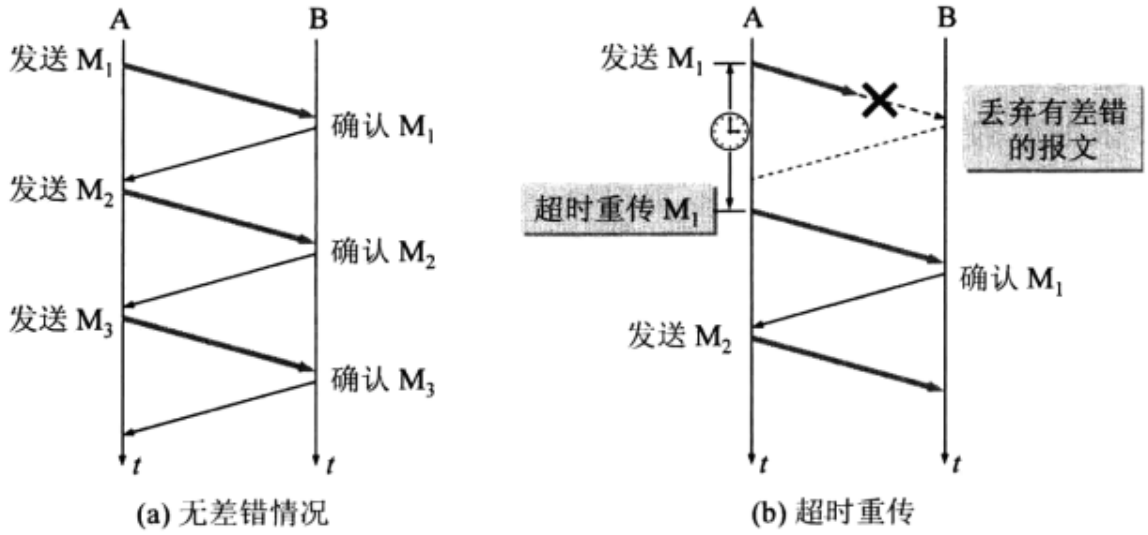


图 5-9　停止等待协议

可靠传输的协议是这样设计的：A只要超过了一段时间仍然没有收到确认，就认为刚才发送的分组丢失了，因而重传前面发送过的分组。这就叫做超时重传。要实现超时重传，就要在每发送完一个分组设置一个超时计时器。如果在超时计时器到期之前收到了对方的确认，就撤销已设置的超时计时器。

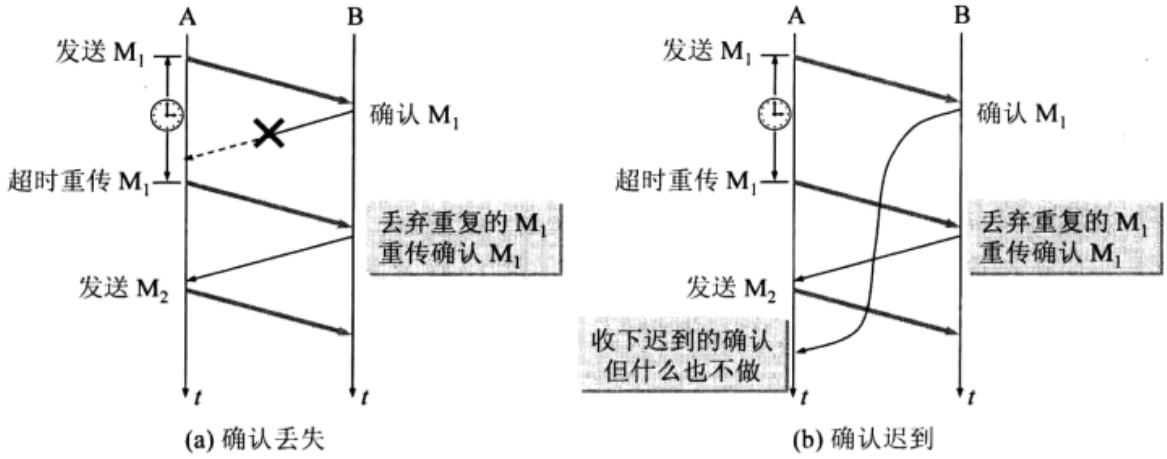


图 5-10　确认丢失和确认迟到

图5-10（a）：B所发送的对M1的确认丢失了，A在设定的超时重传时间内没有收到确认，但并无法知道自己发送的分组出错、丢失，或者是B发送的确认丢失了。因此A在超时计时器到期后就要重传M_1。现在应注意B的动作。假定B又收到了重传的分组M_1。这时采取两个行动：

- 丢弃这个重复的分组M_1，不向上层交付
- 向A发送确认

图5-10（b）也是一种可能出现的情况。传输过程中没有出现差错，但B对分组M_1的确认迟到了。A会收到重复的确认。对重复的确认处理很简单：收下后抛弃。B仍然会收到重复的M_1，并且同样要丢弃重复的M_1，并重传确认分组。

使用上述的确认和重传机制，我们就可以在不可靠的传输网络上实现可靠的通信。像上述的这种可靠传输协议常称为自动重传请求ARQ（Automatic Repeat reQuest）。

2）连续ARQ协议

为了提高传输效率，发送方可以不使用低效率的停止等待协议，而是采用流水线传输。流水线传输就是发送方可以连续发送多个分组，不比每次发完一个分组就停顿下来等待对方的确认。这样可使信道上一直有数据不间断地传送。显然，这种传输方式可以获得很高的信道利用率。

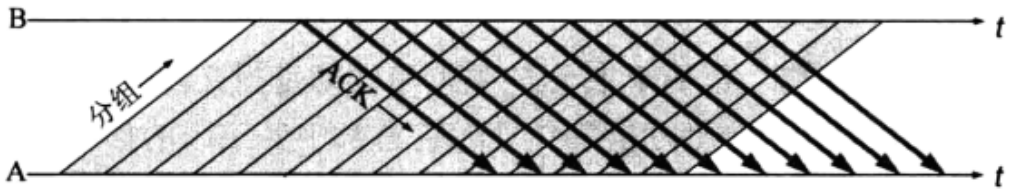


图 5-12　流水线传输可提高信道利用率

当使用流水线传输时，就要使用**连续ARQ协议和滑动窗口协议**。

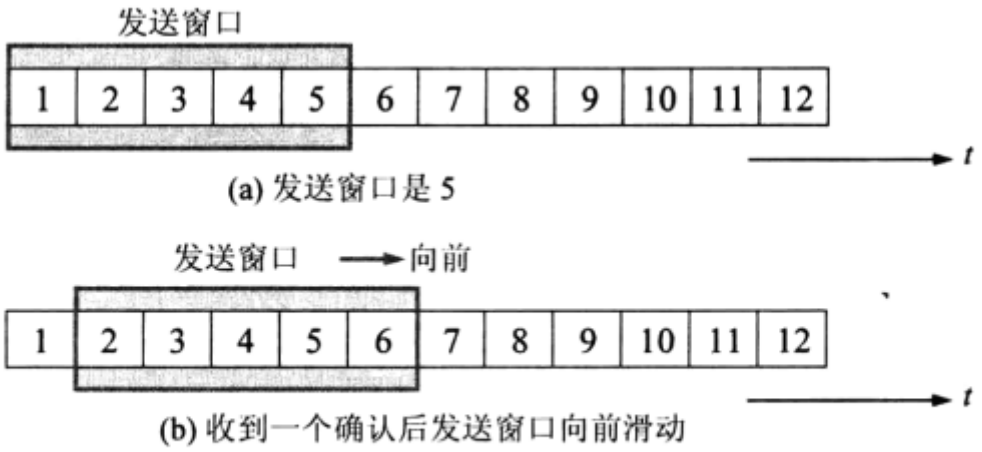


图 5-13　连续 ARQ 协议的工作原理

图5-13（a）表示发送方维持的发送窗口，它的意义是：唯一发送窗口内的5个分组都可以连续发送出去，而不需要等待对方的确认，这样，信道利用率就提高了。连续ARQ协议规定，发送方每接收一个确认，就把发送窗口向前滑动一个分组的位置。图5-13（b）表示发送方收到了对第1个分组的确认，于是把发送窗口向前移动一个分组的位置。如果原来已经发送了5个分组，那么现在就可以发送窗口内的第6个分组了。

接收方一般都是采用累积确认的方式。这就是说，接收方不必对接收到的分组逐个发送确认，而是在收到几个分组后，对按序到达的最后一个分组发送确认，这就表示：到这个分组位置的所有分组都已经正确收到了。

累积确认有优点也有缺点。优点是：容易实现，即使确认丢失也不必重传。但缺点是不能向发送方反映出接收方已经正确收到的所有分组的信息。

例如，如果发送方发送了前5个分组，而中间的第3个分组丢失了。这时接收方只能对前两个分组发出确认。发送方无法知道后面三个分组的下落，而只好把后面的三分分组都再重传一次。这就叫做Go-back-N（回退N），表示需要再退回来重传已经发送过的N个分组。可见通信线路质量不好时，连续ARQ协议会带来负面的影响。

(2) TCP报文段

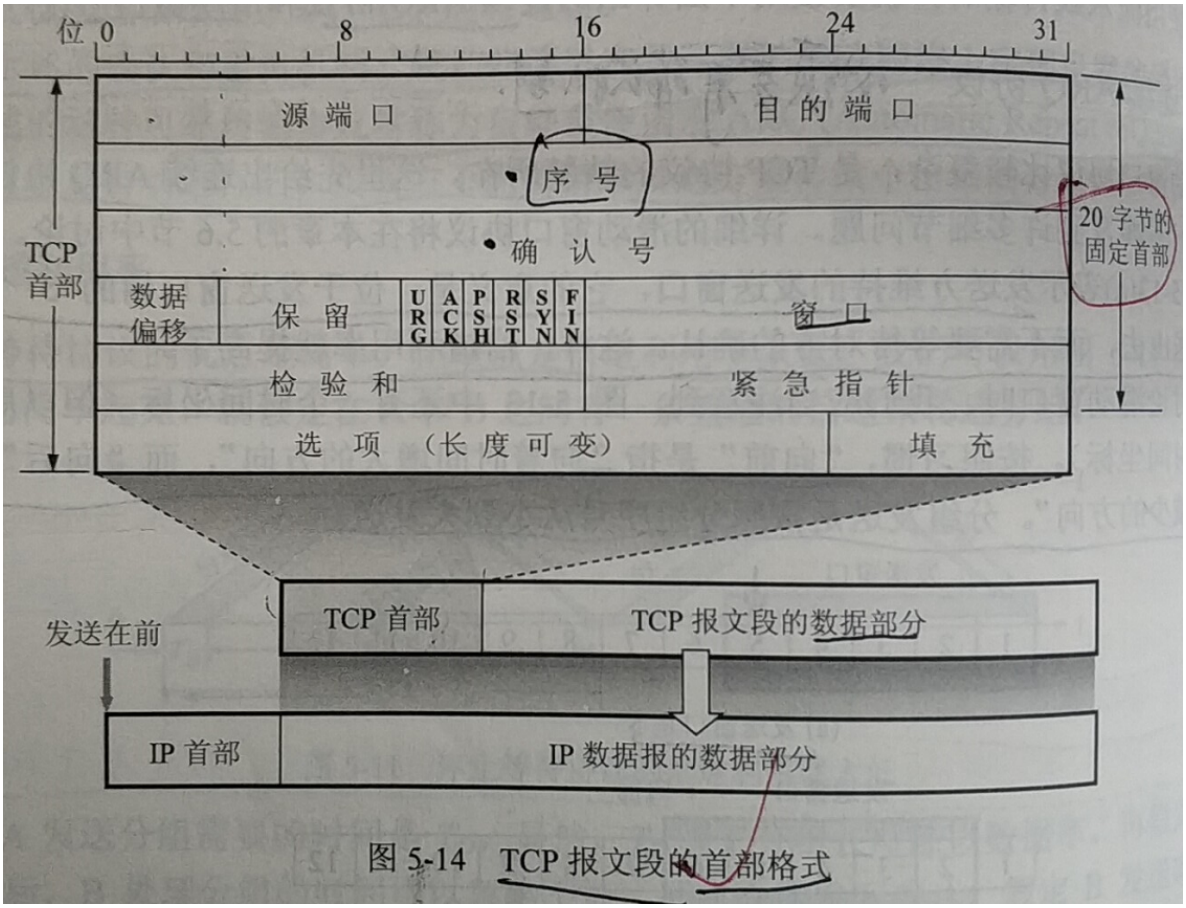


图 5-14 TCP 报文段的首部格式

TCP虽然是面向字节流的，但TCP传送的数据单元却是报文段。TCP报文段首部的前20个字节是固定的，后面有4n字节是根据需要而增加的选项（n是整数）。因此TCP首部的最小长度是20字节。

各个字段的意义：

- **源端口和目的端口**：各占2个字节。TCP的分用功能也是通过端口实现的
- **序号**：占4个字节，序号范围是$[0,2^{32}-1]$，共2^{32}个序号。序号使用mod 2^{32}运算。在一个TCP连接中传送的字节流中的每一个字节都按照顺序编号。首部中的序号字段值则指的是本报文段所发送的数据的第一个字节的序号。
- **确认号**：占4个字节，是期望收到对方下一个报文段的第一个数据字节的序号。例如，B正确收到了A发送过来的一个报文段，其序号字段值是501，而数据长度是200字节（序号501~700），这表明B正确收到了A发送的到序号700为止的数据。因此，B期望收到A的下一个数据序号是701，于是B在发送给A的确认报文段中把确认号置为701.请注意，现在的确认号不是501，也不是700，而是701。
- **数据偏移**：占4位，它指出TCP报文段首部的长度。“数据偏移”的单位是32字（即4字节长的字为计算单位）。由于4位二进制数能够表达的最大十进制数字是15，因此数据偏移的最大值是60字节，这也是TCP首部的最大长度（即选项长度不能超过40字节）。
- **保留**：占6位，保留为今后使用，但目前应置为0
- **紧急URG（URGent）**：当URG=1时，表明紧急指针字段有效，它告诉系统此报文段中有紧急数据，应尽快传送（相当于高优先级的数据）。
- **确认ACK（ACKnowledge）**：仅当ACK=1时确认号才有效。当ACK=0时，确认号无效。TCP规定，在连接简历后所有传送的报文段都必须把ACK置1.
- **推送PSH（PuSH）**：当两个应用进程进行交互式的通信时，有时在一端的应用进程希望在键入一个命令后立即就能收到对方的响应。
- **复位RST（ReSeT）**：当RST=1时，表明TCP连接中出现严重差错（如由于主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接。RST置1还用来拒绝一个非法的报文段或拒绝打开一个连接。

- **同步SYN（SYNchronization）**：在连接建立时用来同步序号。当SYN=1时而ACK=0时，表明这是一个连接请求报文段。对方若同意建立连接，则应在响应的报文段中使SYN=1和ACK=1.因此，SYN置为1就表示这是一个连接请求或连接接收报文。
- **终止FIN**：用来释放一个连接。当FIN=1时，表明此报文段的发送方的数据已发送完毕，并要求释放运输连接。
- **窗口**：占2个字节。窗口值是$[0,2^{16}-1]$之间的整数。窗口指的是发送本报文段的一方的接收窗口（而不是自己的发送窗口）。窗口值作为接收方让发送方设置其发送窗口的依据。
- **检验和**：占2个字节。检验和字段检验的范围包括首部和数据这两部分。
- **紧急指针**：占2个字节，紧急指针仅在URG=1时才有意义。
- **选项**：长度可变，最大可达40字节。

(3) TCP的连接建立

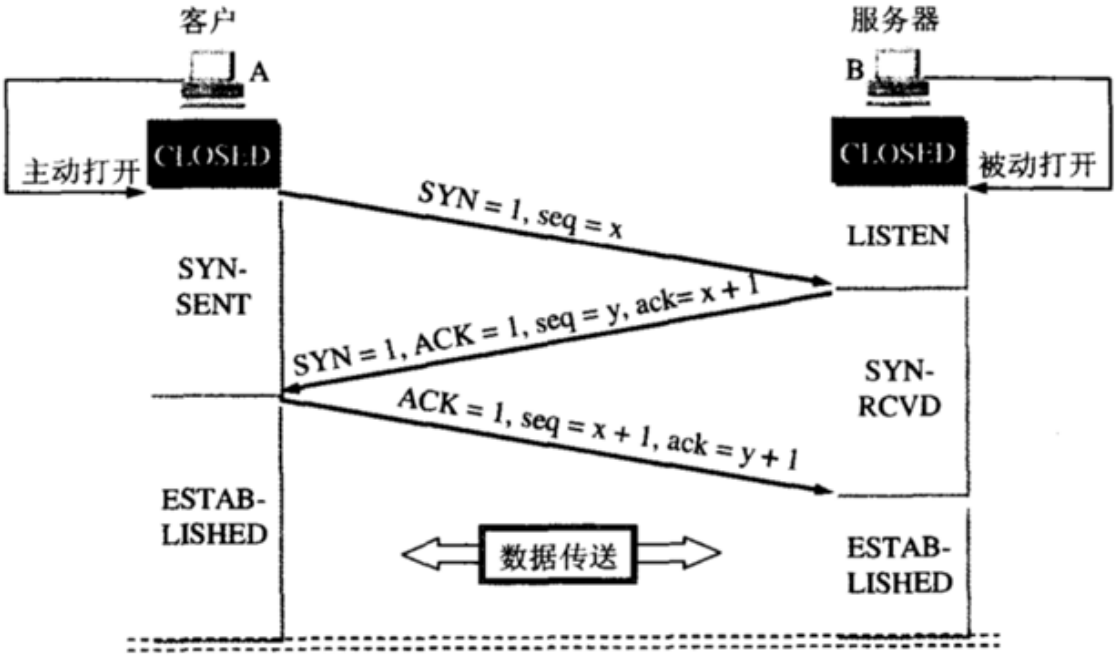


图 5-31　用三次握手建立 TCP 连接

假定主机A运行的是TCP客户程序，而B运行TCP服务器程序。最初两端的TCP进程都处于CLOSED（关闭）状态。图中方框分别是TCP进程所处的状态。A主动打开连接，B被动打开连接。

1. B的TCP服务器进程先创建传输控制块TCB，准备接受客户进程的连接请求。然后服务器进程就处于LISTEN（收听）状态，等待客户的连接请求。如有，即做出响应。
2. A的TCP客户进程也是首先创建传输控制块TCB，然后向B发出连接请求报文段。这时首部中的同步位SYN=1，同时选择一个初始序号seq=x。TCP规定，SYN报文段（即SYN=1的报文段）不能携带数据，但要消耗掉一个序号。这时，TCP客户端进程进入SYN-SENT（同步已发送）状态。
3. B收到连接请求报文段后，如同意建立连接，则向A发送确认。在确认报文段中应把SYN和ACK位都置为1，确认号是ack=x+1，同时也为自己选择一个初始序号seq=y。请注意，这个报文段也不能携带数据，但同样要消耗一个序号。这是TCP服务器进程进入SYN-RCVD（同步收到）状态。
4. TCP客户端进程收到B的确认后，还要向B给出确认。确认报文段的ACK置1，确认号ack=y+1，而自己的序号seq=x+1。TCP的标准规定，ACK报文段可以携带数据。但如果不携带数据则不消耗序号，在这种情况下，下一个数据报文段的序号仍是seq=x+1。这时TCP连接已经建立，A进入ESTABLISHED（已建立连接）状态。
5. 当B收到A的确认后，也进入ESTABLISHED状态。

为什么A还要发送一次确认呢？这主要是为了防止已失效的连接请求报文段突然又传送到B，因而发生错误。

(4) TCP连接释放

数据传输结束后，通信的双方都可释放连接。现在A和B都处于ESTABLISHED状态。

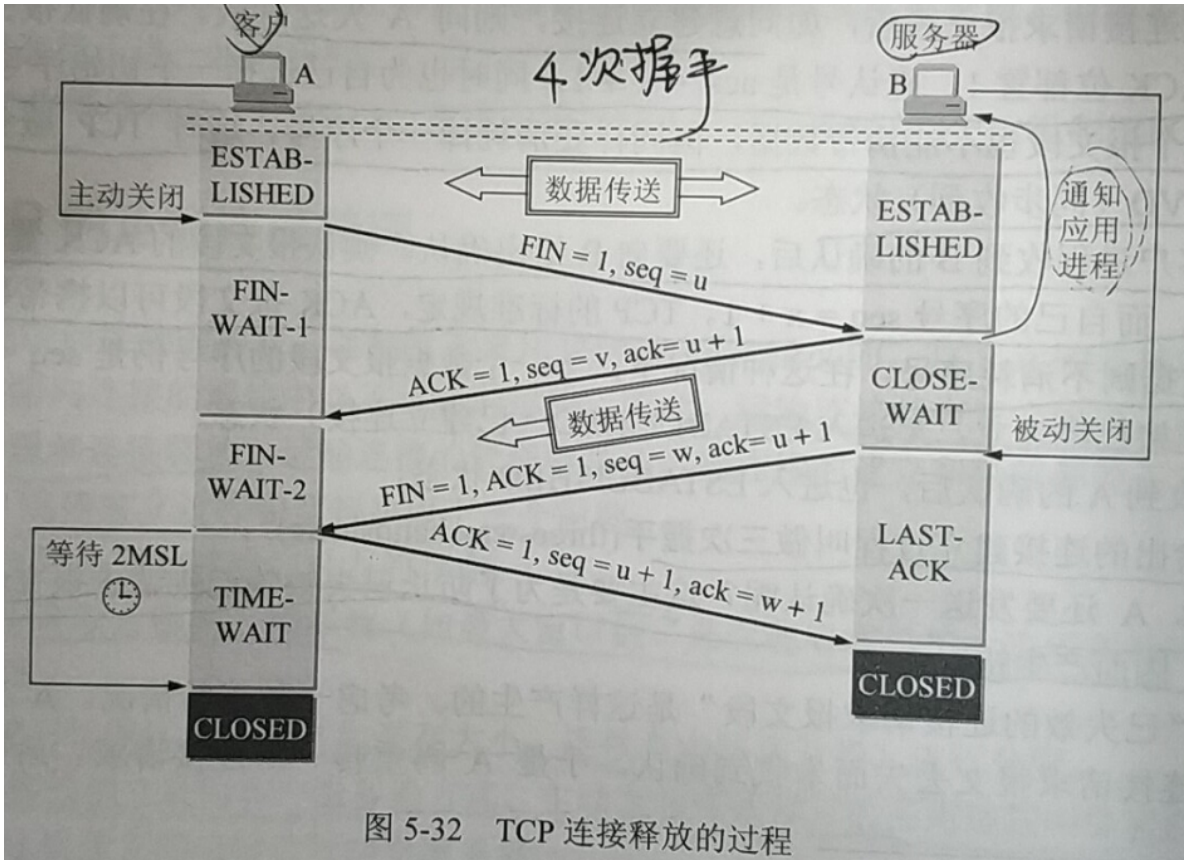


图 5-32　TCP 连接释放的过程

1. A的应用进程先向其TCP发出连接释放报文段，并停止再发送数据，主动关闭TCP连接。A把连接释放报文段首部的终止控制位FIN置1，其序号seq=u，它等于前面已传送过的数据的最后一个字节的序号加1。这时A进入FIN-WAIT-1（终止等待1）状态，等待B的确认。TCP规定，FIN报文段即使不携带数据，它也消耗一个序号。
2. B收到连接释放报文段后即发出确认。确认号是ack=u+1，而这个报文段自己的序号是v，等于B前面已传送过的数据的最后一个字节的序号加1。然后B就进入CLOSE-WAIT（关闭等待）状态。TCP服务器进程这时应通知高层应用进程，因而从A到B这个方向的连接就释放了，这时的TCP连接处于半关闭（half-close）状态，即A已经没有数据要发送了，但B若发送数据，A仍要接收。也就是说，从B到A这个方向的连接并未关闭，这个状态可能会持续一些时间。
3. A收到来自B的确认后，就进入FIN-WAIT-2（终止等待2）状态，等待B发出的连接释放报文段。
4. 若B已经没有要向A发送的数据，其应用进程就通知TCP释放连接。这时B发出的连接释放报文段必须使FIN=1。现假定B的序号为w（在半关闭状态B可能又发送了一些数据）。B还必须重复上次已发送过的确认号ack=u+1。这时B就进入LAST-ACK（最后确认）状态，等待A的确认。
5. A在收到B的连接释放报文段后，必须对此发出确认。在确认报文段中把ACK置1，确认号ack=w+1，而自己的序号是seq=u+1（根据TCP标准，前面发送过的FIN报文段要消耗一个序号）。然后进入到TIME-WAIT（时间等待）状态。请注意，现在TCP连接还没有释放掉。必须进过时间等待计时器（TIME-WAIT timer）设置的时间2MSL后，A才进入到CLOSED状态。时间MSL叫做最长报文段寿命（Maximum Segment Lifetime），MSL可以根据实际情况设置值，RFC 793建议设为2分钟，此时，从A进入到TIME-WAIT状态后，要经过4分钟才能进入到CLOSED状态，才能开始建立下一个新的连接。当A撤销响应的传输控制块TCB后，就结束了这次的TCP连接。

6. B只要收到了A发出的确认，就进入CLOSED状态。同样，B在撤销相应的传输控制块TCB后，就结束了这次TCP连接。我们注意到，B结束TCP连接的时间要比A早一点。
7. TCP还设有一个保活计时器（keepalive timer）。设想有这样的情况：客户已主动与服务器建立了TCP连接。但后来客户端的主机突然出现故障。显然，服务器以后就不能再接收到客户发来的数据。因此，应有措施使服务器不要再白白等待下去。这就是使用保活计时器。服务端每收到一个客户的数据，就重新设置保活计时器，时间的设置通常是两小时。若两个小时没有收到客户的数据，服务器就发送一个探测报文段，以后则每隔75分钟发送一次。若一连发送10个探测报文段后仍无客户的响应，服务器就认为客户端出了故障，接着就关闭这个链接。

为何A在TIME-WAIT状态必须等待2MSL的时间呢？有两个理由：

1. 为了保证A发送的最后一个ACK报文段能够到达B。这个报文段有可能丢失，因而使处于LAST-LOCK状态的B收不到对方已发送的FIN+ACK报文段的确认。B会超时重传这个FIN+ACK报文段，而A就能在2MSL的时间内收到这个重传的FIN+ACK报文段，而A就能在2MSL时间内收到这个重传的FIN+ACK报文段。
2. 防止“已失效的连接请求报文段”出现在本连接中。A在发送完最后一个ACK报文段后，再经过时间2MSL，就可以使本连接持续的时间内所产生的所有报文段都从网络上小时，这样就可以使下一个新的连接中不会出现这种旧的连接请求报文段。

除时间等待计时器外，TCP还设有一个保活计时器。设想有这样的情况：客户已经主动与服务器建立了TCP连接。因此，应当有措施使服务器不要再白白等待下去。这就是使用保活计时器。服务器每收到一次客户的数据，就重新设置保活计时器，时间的设置通常是两个小时。若两小时没有收到客户的数据，服务器就发送一个探测报文段，以后每隔75分钟发送一次。若一连发送10个探测报文段后仍无客户的响应，服务器就认为客户端出了故障，接着就关闭这个连接。

四、Http、Https和TCP/IP

1.HTTP/1.0在用户点击鼠标后发生的几个事件

（1）浏览器分析连接指向页面的URL

（2）浏览器向DNS请求解析域名的IP地址

（3）域名系统DNS解析出对应的IP地址

（4）浏览器与服务器简历TCP连接

（5）浏览器发出命令

（6）服务器做出响应，将结果返回给浏览器

（7）释放TCP连接

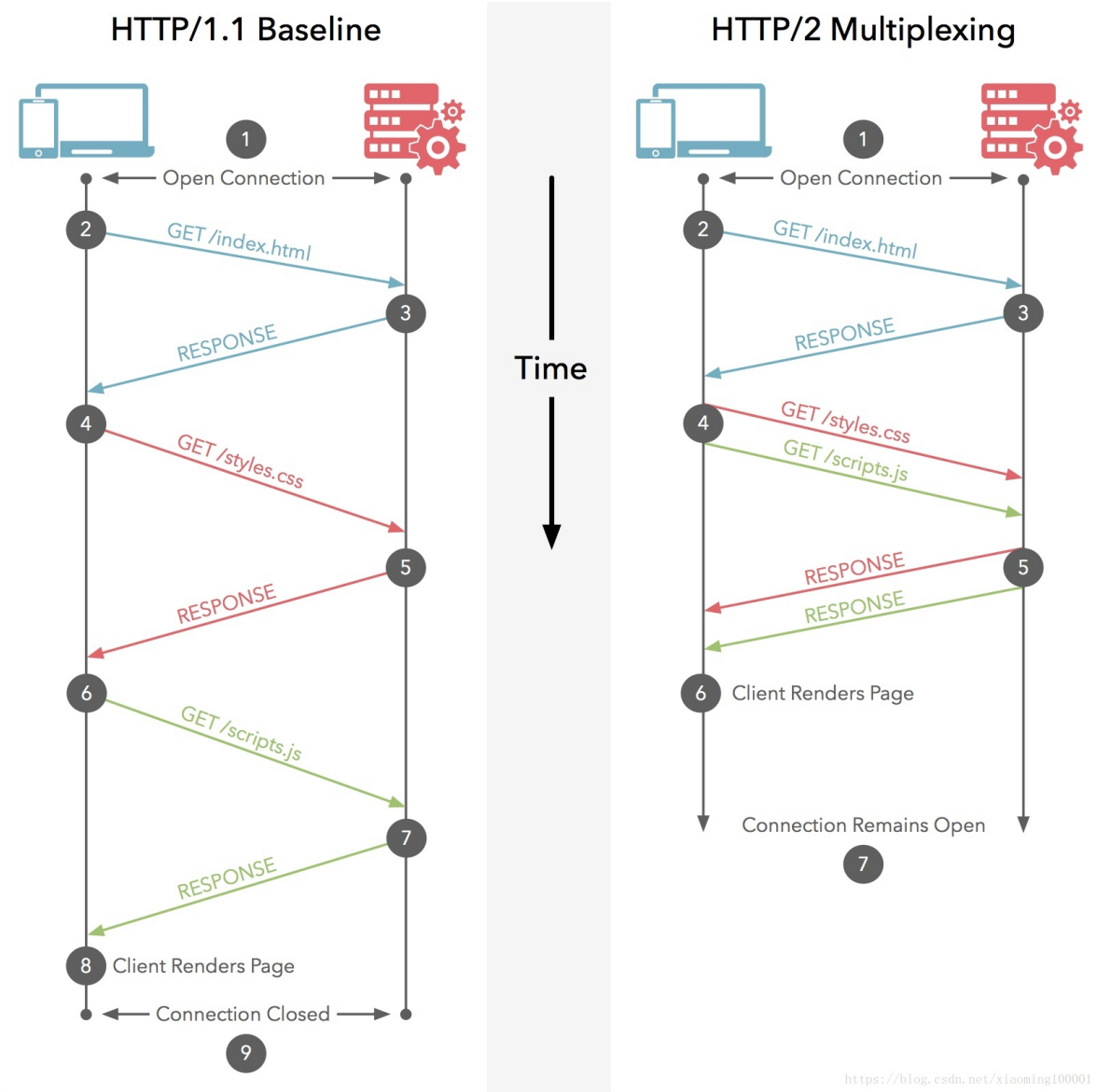
（8）浏览器展示结果

HTTP使用了面向连接的TCP作为运输层协议，保证了数据的可靠传输。HTTP不必考虑数据在传输过程中被丢弃后又怎样被重传。但是，HTTP协议本身是无连接的。虽然HTTP使用了TCP连接，但通信的双方在交换HTTP报文之前不需要先建立HTTP连接。

HTTP协议时无状态的。同一个客户第二次访问同一服务器上的页面时，服务器的响应与第一次被访问时的相同，因为服务器并不记得曾经访问过的这个客户。

2.HTTP 1.0和HTTP 1.1的对比

版本	产生时间	内容	发展现状
HTTP/0.9	1991年	不涉及数据包传输，规定客户端和服务器之间通信格式，只能GET请求	没有作为正式的标准
HTTP/1.0	1996年	传输内容格式不限制，增加PUT、PATCH、HEAD、 OPTIONS、DELETE命令	正式作为标准
HTTP/1.1	1997年	持久连接(长连接)、节约带宽、HOST域、管道机制、分块传输编码	2015年前使用最广泛
HTTP/2	2015年	多路复用、服务器推送、头信息压缩、二进制协议等	逐渐覆盖市场



HTTP/1.0的主要缺点，就是每请求一个文档就要有两倍RTT的开销，特别是万维网服务器往往要同时服务于大量客户的请求，所以这种非持续连接会使万维网服务器负担很重。

HTTP/1.1协议很好地解决了这个问题，它使用了持续连接。所谓持续连接就是万维网服务器在发送响应后仍然在一段时间内保持这条连接，使同一个客户和该服务器可以持续在这条连接上传送后续的HTTP请求报文和响应报文。

HTTP/1.1协议的持续连接有两种工作方式，即非流水线方式和流水线方式。

流水线方式的特点，是客户在收到前一个响应后才能发出下一个请求。因此，在TCP连接已建立后，客户每访问一次对象都要用去一个往返时间RTT。这比非持续连接要用去两倍RTT的开销，节省了建立TCP连接所需的一个RTT时间。但非流水线方式还是有缺点的，因为服务器在发送完一个对象后，其TCP连接就处于空闲状态，浪费了服务器资源。

流水线方式的特点，是客户在收到HTTP的响应报文之前就能够接着发送新的请求报文。于是一个接一个的请求报文到达服务器后，服务器可持续发回响应报文。因此，使用流水线方式时，客户访问所有的对象只需要花费一个RTT时间。流水线工作方式使TCP连接中的空闲时间减少，提高了下载文档效率。

3.HTTP状态码

- 1XX表示通知信息，如请求收到了或正在进行处理
- 2XX表示成功，如接受或知道了
- 3XX表示重定向，如要完成请求还必须采取进一步的行动
- 4XX表示客户的差错，如请求中错误的语法或不能完成
- 5XX表示服务器的差错，如服务器失效或无法完成请求

4.安全问题

（1）无线WIFI网络的攻击

攻击者会提供免费的WiFi服务，一旦连接上恶意的WiFi网络，用户毫无隐私，尤其对于基于HTTP的应用来说。提供WiFi网络的攻击者可以截获所有的HTTP流量，HTTP流量本身是明文的，攻击者可以很轻松的获取用户的详细信息，造成信息泄露，这种攻击方式也被称为被动攻击。

（2）垃圾广告攻击

很多用户在浏览某个网页时，经常发现页面上弹出一些毫不相干的广告，这种攻击很常见，主要是ISP（互联网服务提供商）发动的一个攻击，用户根本没有任何办法防护。

4.SSL使用的加密算法

SSL全名Security Sockets Layer，即安全套接层，它是为网络通信提供安全及数据完整性的一种安全协议，是操作系统对外的API，SSL3.0后更名为TLS。它采用身份验证和数据加密保证网络通信的安全和数据的完整性。

（1）对称加密

对称加密是加密和解密使用同一密钥。其工作过程是，发送方使用密钥将明文数据加密成密文，然后发送出去，接收方收到密文后，使用同一个密钥将密文解密成明文读取。

优点：加密计算量小，速度快，适合对大量数据进行加密的场景使用。

缺点：

- 密钥传输问题：如上所说，由于对称加密的加密和解密使用的是同一密钥，所以对称加密的安全性不仅仅取决于加密算法本身的强度，更取决于密钥是否被安全的保管，因此加密者如何把密钥安全的传递到解密者手里，就成了对称加密面临的关键问题。（比如，我们客户端肯定不能直接存储对称加密的密钥，因为被反编译之后，密钥就泄露了，数据安全性就得不到保障，所以实际中我们一般都是客户端向服务端请求对称加密的密钥，而且密钥还得用非对称加密加密后再传输。）

- 密钥管理问题：再者随着密钥数量的增多，密钥的管理问题会逐渐显现出来。比如我们在加密用户的信息时，不可能所有用户都用同一个密钥加密解密吧，这样的话，一旦密钥泄漏，就相当于泄露了所有用户的信息，因此需要为每一个用户单独的生成一个密钥并且管理，这样密钥管理的代价也会非常大。

典型的加密算法有：DES、AES、RC4等。

（2）非对称加密

非对称加密是加密和解密的密钥不相同，即公钥和私钥。加密和解密可以使用不同的规则，只要两种规则之间存在某种对应关系即可，这样避免了直接传递私钥。

- 乙方生成两把密钥（公钥和私钥）。公钥是公开的，任何人可以获取，私钥则是保密的。
- 甲方获取乙方的公钥，然后用它对信息加密。
- 乙方得到加密信息，用私钥解密。

如果公钥加密的信息只有私钥解得开，那么只要私钥不泄露，通信就是安全的。即RSA算法。密钥越长，它就越难破解。非对称加密速度较慢。

典型的加密算法有：RSA、DSA、DSS等

（3）哈希算法

哈希算法是将任意长度的信息转换成固定长度的值，不可逆，它常用于验证数据的完整性。它具有如下特点：

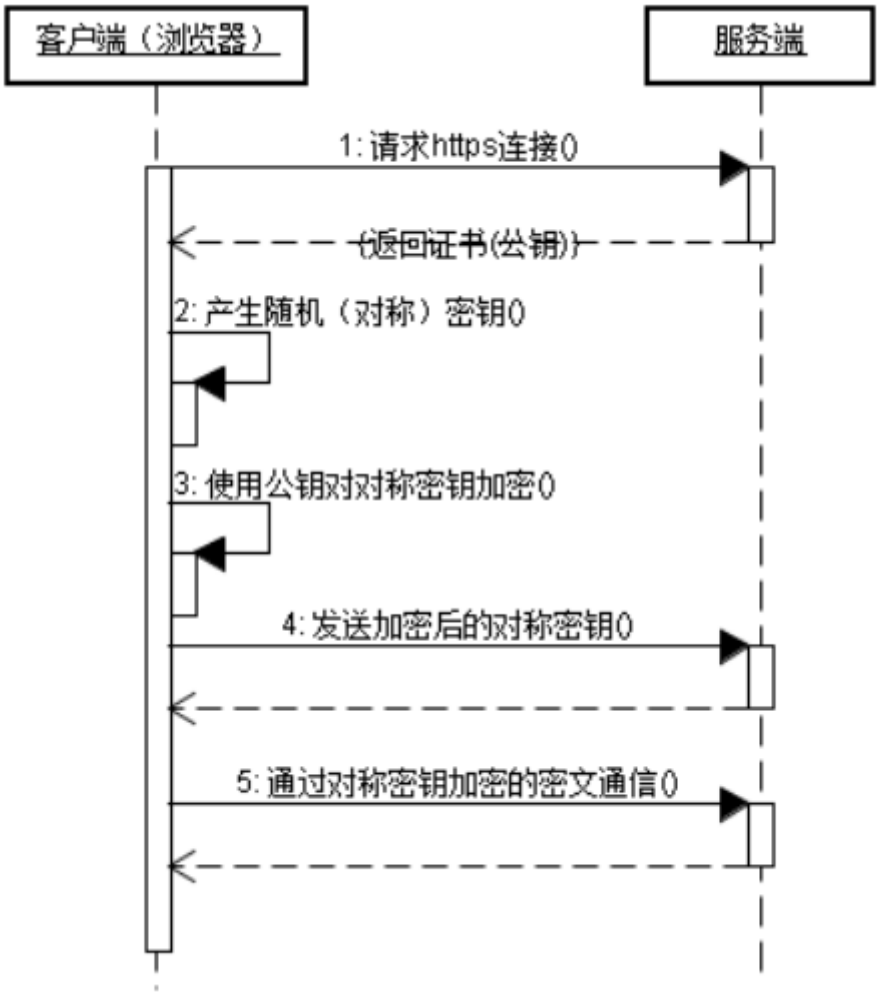
- 信息相同，字符串也相同
- 信息相似，不会影响字符串相同
- 可以生成无数的信息，但字符串的种类是一定的，所以是不可逆的

典型的Hash算法有：MD5

（4）数字签名

数字签名是保证某个信息或文件是某人发出或认可的。

5.HTTPS请求



https是一种加密传输协议，基于非对称加密算法和对称加密算法的协作使用。

其他、面经

1.浏览器通过URL地址请求做了哪些事?

1、域名解析：浏览器获得URL地址，向操作系统请求该URL对应的IP地址，操作系统查询DNS（首先查询本地HOST文件，没有则查询网络）获得对应的IP地址

解释：

把URL分割成几个部分：协议、网络地址、资源路径

协议：指从该计算机获取资源的方式，常见的是HTTP、FTP

网络地址：可以是域名或者是IP地址，也可以包括端口号，如果不注明端口号，默认是80端口

如果地址不是一个IP地址，则需要通过DNS（域名系统）将该地址解析成IP地址，IP地址对应着网络上的一台计算机，DNS服务器本身也有IP，你的网络设置包含DNS服务器的IP，例如，www.abc.com不是一个IP，则需要向DNS询问请求www.abc.com对应的IP，获得IP，在这个过程中，你的电脑直接询问DNS服务器可能没有发现www.abc.com对应的IP，就会向它的上级服务器询问，这样依次一层层向上级找，最高可达根节点，直到找到或者全部找不到为止

端口号就相当于银行的窗口，不同的窗口负责不同的服务，如果输入www.abc.com:8080/，则表示不使用默认的80端口，而使用指定的8080端口

2、确认好了IP和端口号，则可以向该IP地址对应的服务器的该端口号发起TCP连接请求

3、服务器接收到TCP连接请求后，回复可以连接请求

4、浏览器收到回传的数据后，还会向服务器发送数据包，表示三次握手结束

5、三次握手成功后，开始通讯，根据HTTP协议的要求，组织一个请求的数据包，里面包含请求的资源路径、你的身份信息等，例如，www.abc.com/images/1/表示的资源路径是images/1/，发送后，服务器响应请求，将数据返回给浏览器，数据可以是根据HTML协议组织的网页，里面包含页面的布局、文字等等，也可以是图片或者脚本程序等，如果资源路径指定的资源不存在，服务器就会返回404错误，如果返回的是一个页面，则根据页面里的一些外链URL地址，重复上述步骤，再次获取

6、渲染页面，并开始响应用户的操作

7、窗口关闭时，浏览器终止与服务器的连接