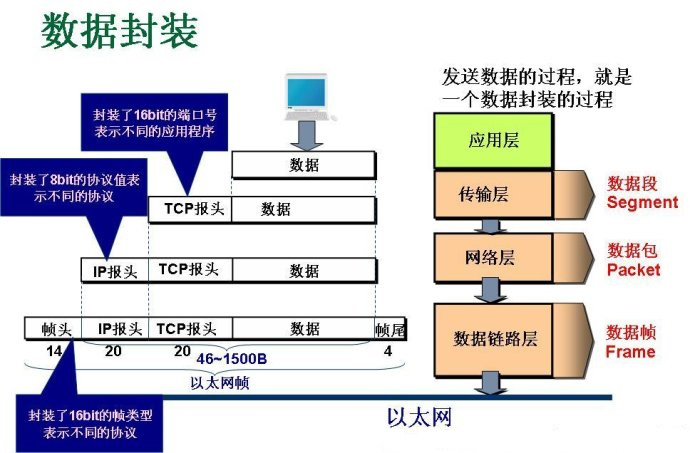
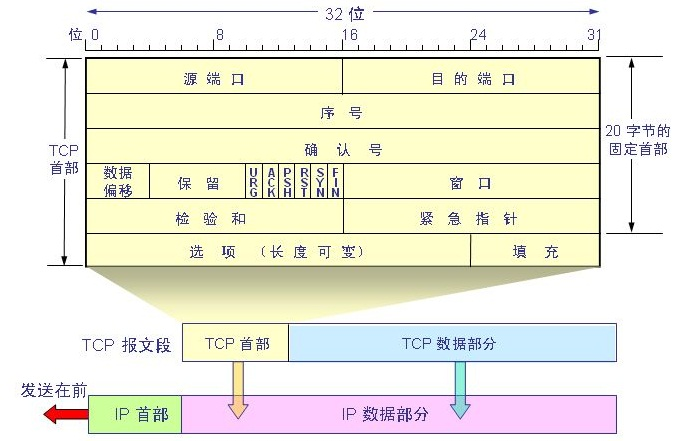
**1. TCP/IP结构**

**TCP/IP数据封装:**

****

**TCP:**

****

**TCP三次握手:**

第一次握手：

客户端发送：（标志位）SYN=1，（SYN序列号，随机生成）Sequence Number=j，（标志位）ACK=0。

第二次握手：

服务器端发送：（标志位）ACK=1，（ACK确认序号）Acknowledgement Number=j+1，（标志位）SYN=1，（SYN序列号，随机生成）Sequence Number=k。

第三次握手：

客户端发送：（标志位）ACK=1，（确认序号）Acknowledgement Number=k+1，（标志位）SYN=0。

**TCP四次挥手:**

第一次挥手：

客户端发送：（标志位）FIN=1，（FIN序列号，随机生成）Finish Number=j，（标志位）ACK=0。

第二次挥手：

服务器端发送：（标志位）ACK=1，（ACK确认序号）Acknowledgement Number=j+1，（标志位）FIN=0。

第三次挥手：

服务器端发送：（标志位）FIN=1，（FIN序列号，随机生成）Sequence Number=k，（标志位）ACK=0。

第四次挥手：

客户端发送：（标志位）ACK=1，（确认序号）Acknowledgement Number=k+1，（标志位）FIN=0。

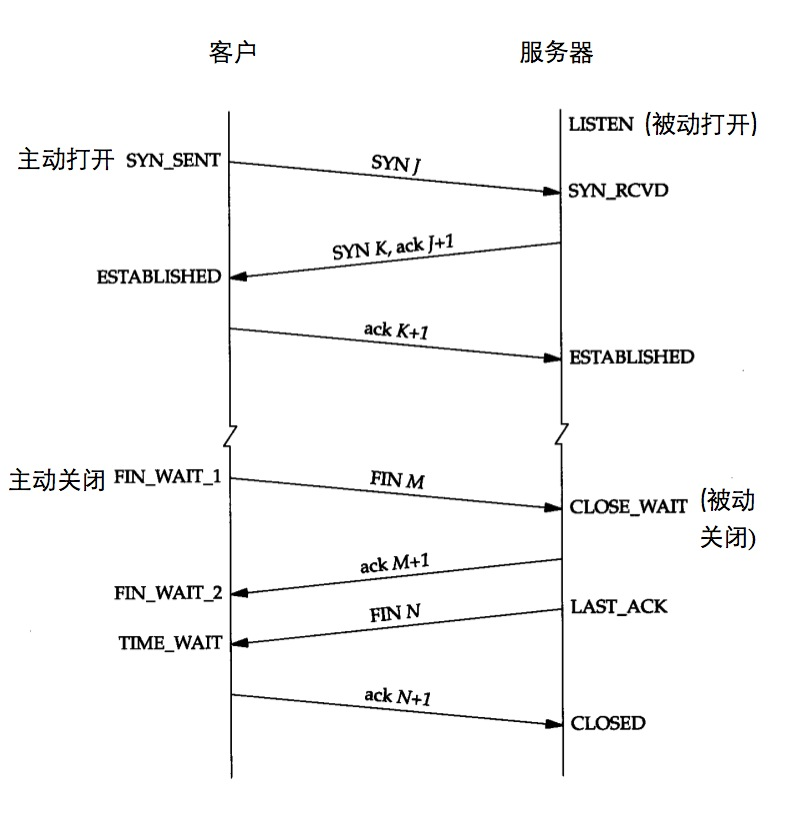
**客户端TCP状态迁移：**

CLOSED → SYN\_SENT → ESTABLISHED → FIN\_WAIT\_1 → FIN\_WAIT\_2 → TIME\_WAIT → CLOSED

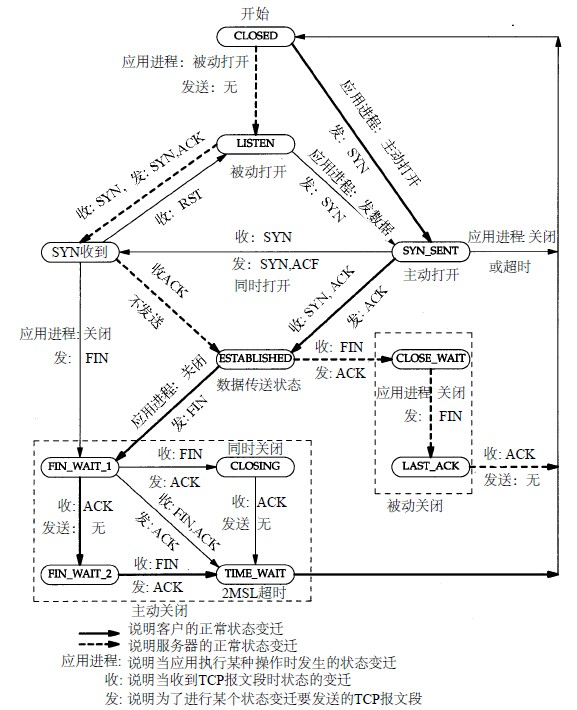
**服务器端TCP状态迁移：**

CLOSED → LISTEN → SYN\_RCVD → CLOSE\_WAIT → LAST\_ACK → CLOSED

**TCP正常连接建立和终止所对应的状态：**



**TCP状态变迁图：**



**2. TCP状态转换伪代码**

TCP\_Main\_Module(segment){

　　查找 TCB（TransmitControlBlock）

　　if(相应的TCB未找到) 创建TCB，其状态为CLOSED

　　找到TCB表中相应表项的状态

　　swith（状态）{

　　　　///

　　　　case CLOSED 状态：

　　　　　　if（收到 被动打开 报文）进入LISTEN状态

　　　　　　if（收到 主动打开 报文）{

　　　　　　　　发送SYN报文段

　　　　　　　　进入SYN\_SENT状态

　　　　　　}

　　　　　　if（收到任何报文段）发送RST报文段

　　　　　　if（收到其他任何报文）发出差错报文

　　　　　　break

　　　　///

　　　　case LISTEN 状态：

　　　　　　if（收到 发送数据 报文）{

　　　　　　　　发送SYN报文段

　　　　　　　　进入SYN\_SENT状态

　　　　　　}

　　　　　　if（收到 任何SYN报文段 ）{

　　　　　　　　发送SYN+ACK报文段

　　　　　　　　进入SYN\_RCVD状态

　　　　　　}

　　　　　　if（收到任何其他报文端或者报文）{

　　　　　　　　发出发错报文

　　　　　　}

　　　　　　break

　　　　///

　　　　case SYN\_SENT 状态：

　　　　　　if（超时）进入CLOSED状态

　　　　　　if（收到SYN报文段）{

　　　　　　　　发送SYN+ACK报文段

　　　　　　　　进入SYN+RCVD状态

　　　　　　}

　　　　　　if（收到SYN+ACK报文段）{

　　　　　　　　发送ACK报文段

　　　　　　　　进入ESTABLISHED状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　case SYN\_RCVD 状态：

　　　　　　if（收到ACK报文）进入ESTABLISH状态

　　　　　　if（超时）{

　　　　　　　　发送RTS报文

　　　　　　　　进入CLOSED状态

　　　　　　}

　　　　　　if(收到 关闭 报文){

　　　　　　　　进入FIN报文段

　　　　　　　　进入FIN\_WAIT1状态

　　　　　　}

　　　　　　if（收到RTS报文段）{

　　　　　　　　进入LISTEN状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　//

　　　　case ESTABLISHED 状态：

　　　　　　if（收到FIN报文段）{

　　　　　　　　发送FIN报文段

　　　　　　　　进入CLOSED-WAIT状态

　　　　　　}

　　　　　　if（收到 关闭 报文）{

　　　　　　　　发送FIN报文段

　　　　　　　　进入FIN-WAIT1状态

　　　　　　}

　　　　　　if（收到RTS或者SYN报文段）发出差错报文

　　　　　　if（收到数据或者ACK报文段）调用输入模块

　　　　　　if（收到 发送 报文）调用输出模块

　　　　　　break

　　　　////

　　　　case FIN-WAIT1 状态：

　　　　　　if（收到FIN报文段）{

　　　　　　　　发送ACK报文段

　　　　　　　　进入CLOSING状态

　　　　　　}

　　　　　　if（收到FIN+ACK报文段）{

　　　　　　　　发送ACK报文段

　　　　　　　　进入TIME-WAIT状态

　　　　　　}

　　　　　　if（收到ACK报文段）{

　　　　　　　　进入FIN-WAIT2状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　////

　　　　case FIN-WAIT2 状态：

　　　　　　if（收到FIN报文段）{

　　　　　　　　发送ACK报文段

　　　　　　　　进入TIME-WAIT状态

　　　　　　}

　　　　　　break

　　　　case CLOSING 状态：

　　　　　　if（收到ACK报文段）{

　　　　　　　　进入TIME-WAIT状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　case TIME-WAIT 状态：

　　　　　　if（超时）{

　　　　　　　　进入CLOSED状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　case CLOSED-WAIT 状态：

　　　　　　if（收到 关闭 报文）{

　　　　　　　　发送FIN报文段

　　　　　　　　进入LAST-ACK状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　case LAST-ACK 状态：

　　　　　　if（收到ACK报文段）{

　　　　　　　　进入CLOSED状态

　　　　　　}

　　　　　　if（收到任何其他报文段或者报文）{

　　　　　　　　发出差错报文

　　　　　　}

　　　　　　break

　　　　///Ending

　　}

}

一个TCP连接由一个四元组唯一确定：本地IP地址，本地端口号，远程IP地址，远程端口号。

第三次握手时客户端要先确认服务器端发来的ACK确认序号是否等于客户端在第一次握手时发送的值加1。

三次握手的中间状态叫做半连接状态，如果握手没有完成，在超时之前，两个主机将一直处于半连接状态

服务器端收到信息如果有错就丢弃，没错就回复确认

客户端如果在规定时间内没有收到确认，就重传

发起中断连接的可以是客户端也可以是服务器端。

任何TCP实现都必须为MSL（最大报文段生存时间，Maximum Segment Lifetime）选择一个值，RFC建议2分钟，Berkeley建议30秒。

TIME\_WAIT状态中所需要的时间是依赖于实现方法的。典型的值为30秒、1分钟和2分钟。等待之后连接正式关闭，并且所有的资源(包括端口号)都被释放。

TIME\_WAIT状态需要经过2MSL才能返回到CLOSE状态：

虽然四个报文都发送完毕，可以直接进入CLOSE状态，但是必须假想网络是不可靠的，有可能最后一个ACK丢失。所以TIME\_WAIT状态就是用来重发可能丢失的ACK报文。

TIME\_WAIT存在有两个理由：

1. 保证TCP全双工连接的终止

2. 使得本次连接的所有分组传输完成，并且使得徘徊的分组在网络中超过MSL时间而消逝。

连接超时：

第一次超时时间：6秒计时器

第二次超时时间：24秒计时器

最大报文段长度（MSS）：

默认536字节，对于以太网MSS最大可达到1460字节，MSS出现在SYN报文段中

TCP提供半关闭状态，此时连接的一端在结束连接后到收到对方FIN前，还能接收另一方的数据。

第二次挥手后可能出现无限等待状态

TCP首部的RST比特是用来复位的，发生情况：

1. 当连接请求到达时，目的端口没有进程在听。

2. 用于异常终止一个连接

同时打开：

当两台主机执行主动打开建立连接时，TCP将仅建立一条连接，一个同时打开的连接需要交换4个报文段。

同时关闭：

同时关闭与正常关闭使用的段交换数目相同。

特殊情况可能产生死锁：

确认信息的丢失可能会引起系统的死锁。当接受方发送了确认，同时把消息的窗口大小调整为0，（请求关闭发送窗口时就会发生这样的情况）。过了一段时间之后，接受方打算取消这一限制，但是如果它没有数据要发送，就会发送确认包，并且利用一个窗口大小非零的数值来取消这个限制。如果这个确认丢失了，就会产生问题，发送方一直在等待确认一个非零的窗口大小，而接收方认为发送方已经收到了这个确认，正在等待数据。双方都在等待一个纯粹的ACK确认，不涉及窗口内报文的确认不能启用RTO机制来重新发送ACK。要避免这种死锁，就要设计一种持续计时器来处理这个问题。

建立Socket连接，客户端执行connect()时触发TCP三次握手连接。服务器端执行accept将已完成的连接从队列（FIFO队列）中取出。

报文丢失，分组在网络中“迷途”通常是路由器异常的结果。当报文分组由于路由器故障在网络中徘徊（迷途的重复分组），并且小于MSL时间。如果发送端对该分组进行超时重传，则接收端可能会收到两个重复的分组。

自定义传输协议时，格式可参考如下定义

数据头+数据长度+数据内容+校验码+数据尾