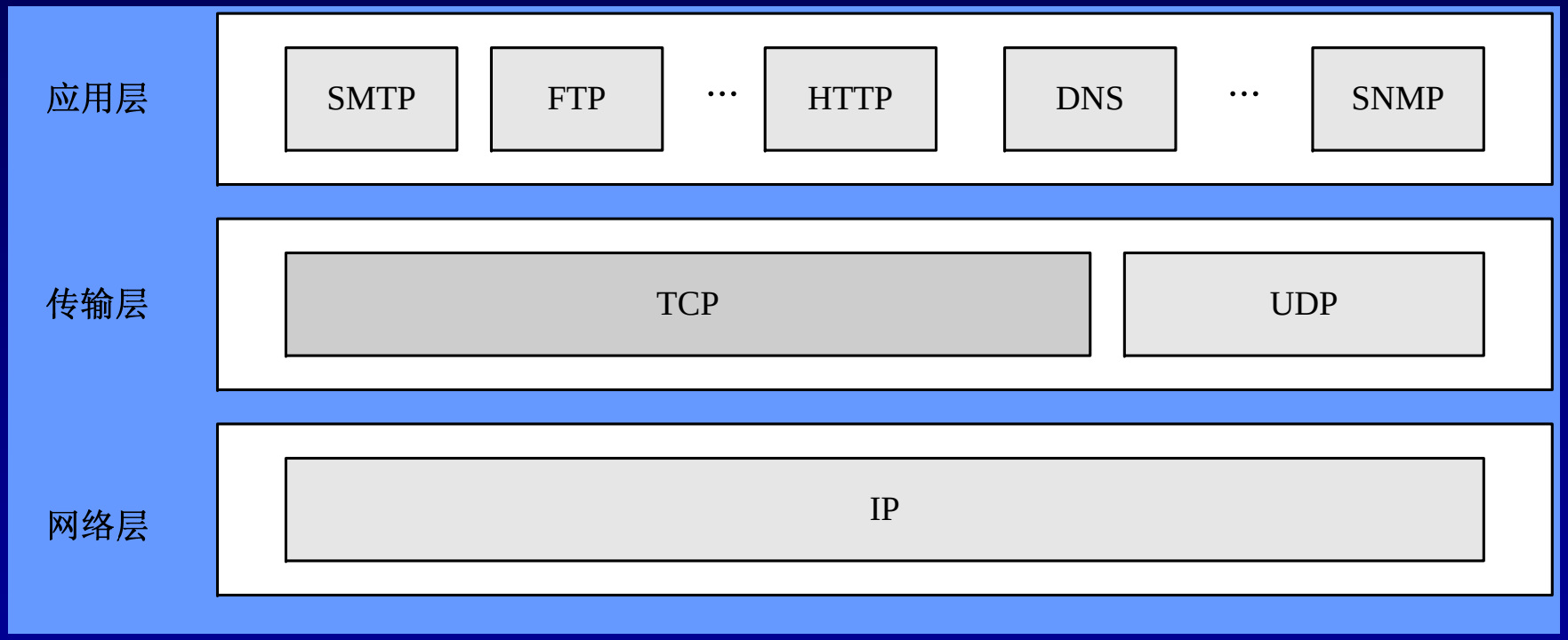
传输层

传输控制协议TCP

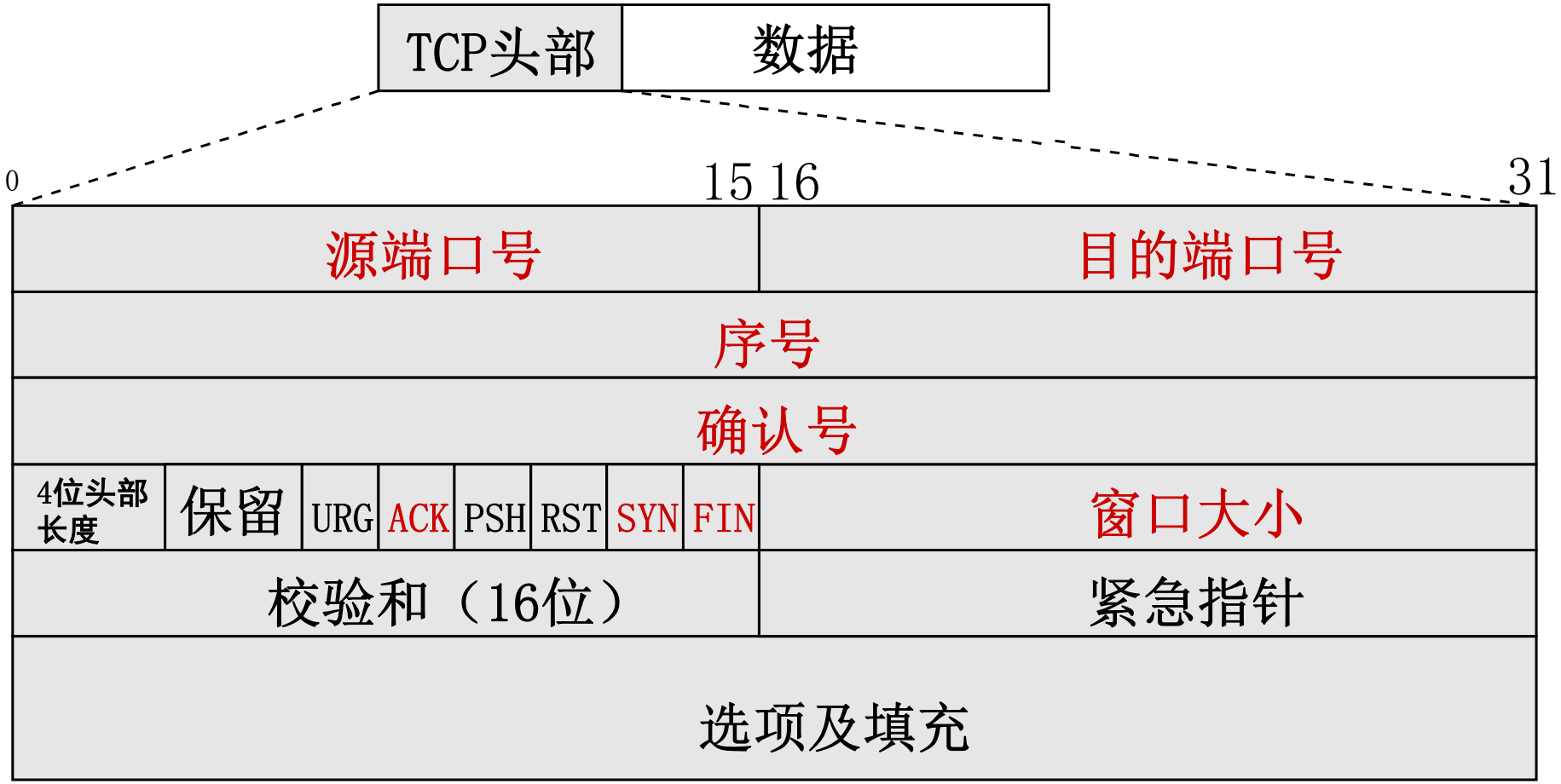
传输控制协议TCP

- ✓ TCP是一种面向连接的、可靠的传输层协议；
- ✓ TCP协议建立在不可靠的网络层IP协议之上，IP不能提供任何可靠性机制，TCP的可靠性完全由自己实现；
- ✓ TCP采用的最基本的可靠性技术是：
  - 确认与超时重传；
  - 滑动窗口机制进行流量控制。

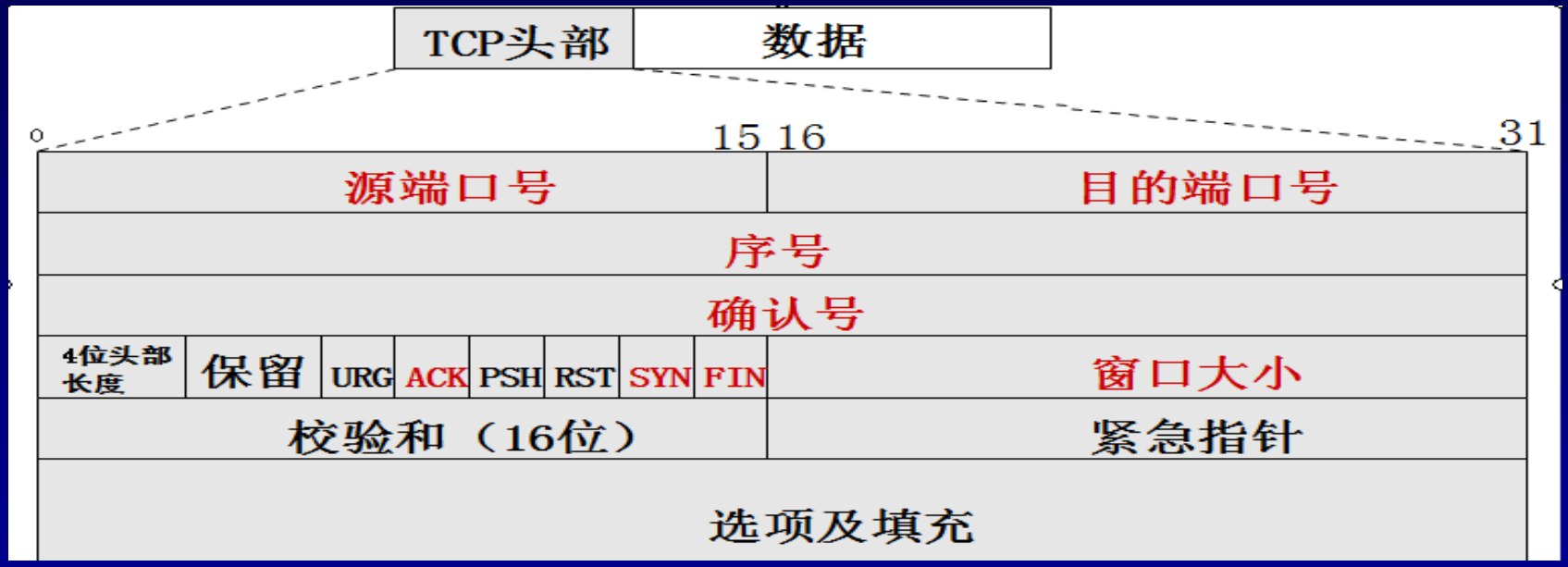
TCP协议与其他协议的层次关系



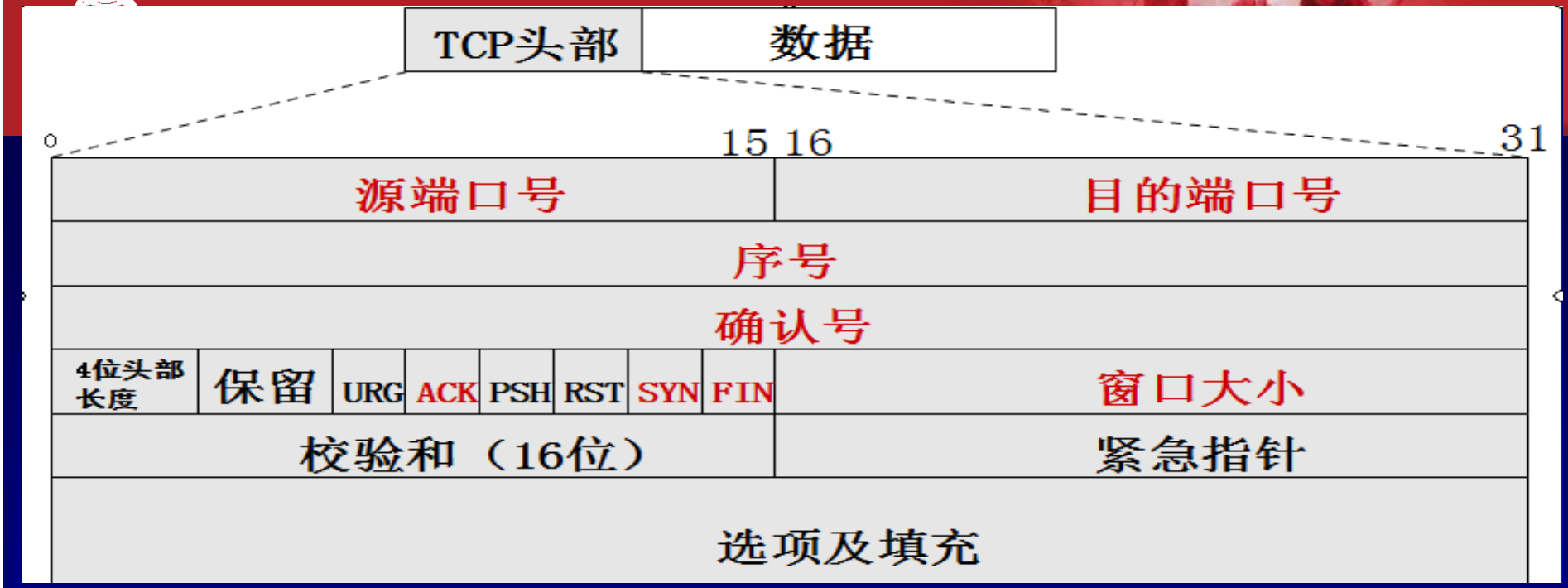
TCP报文格式


TCP头部
数据
0
15 16
31
源端口号
目的端口号
序号
确认号
4位头部长度
保留
URG
ACK
PSH
RST
SYN
FIN
窗口大小
校验和（16位）
紧急指针
选项及填充

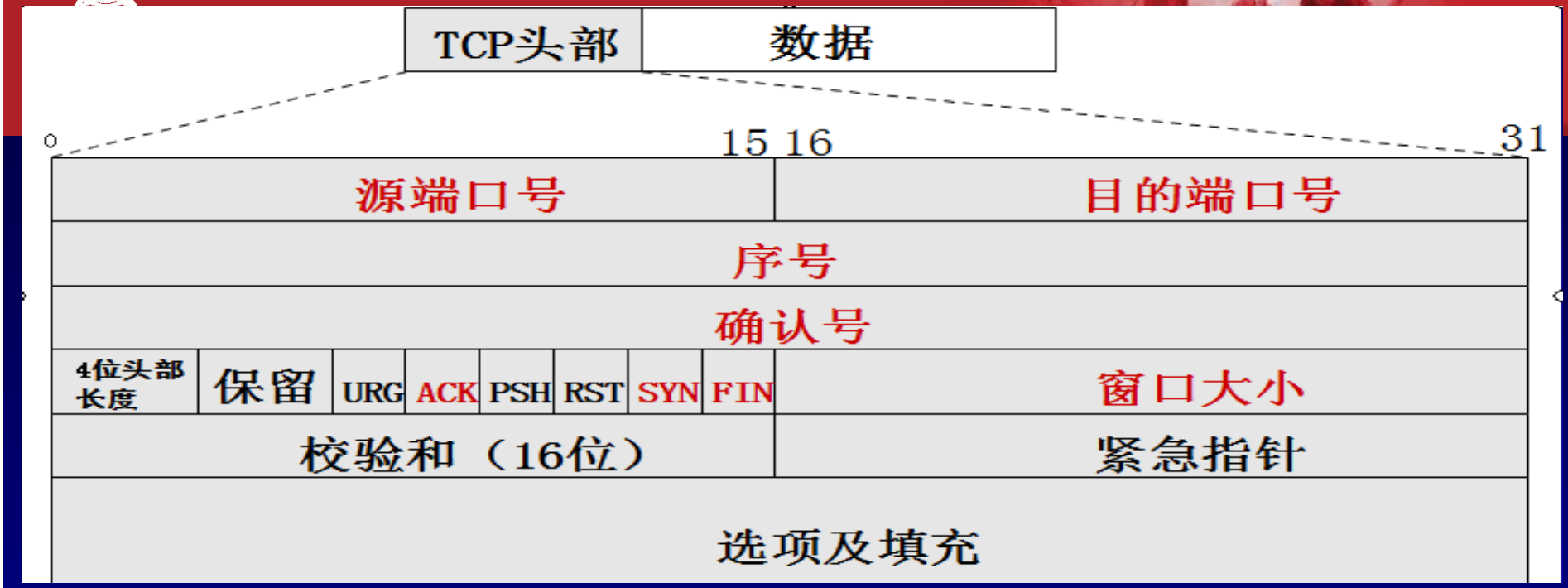
首部固定部分各字段的意义



1 源端口和目的端口　　各占2个字节。端口是传输层与高层的服务接口。

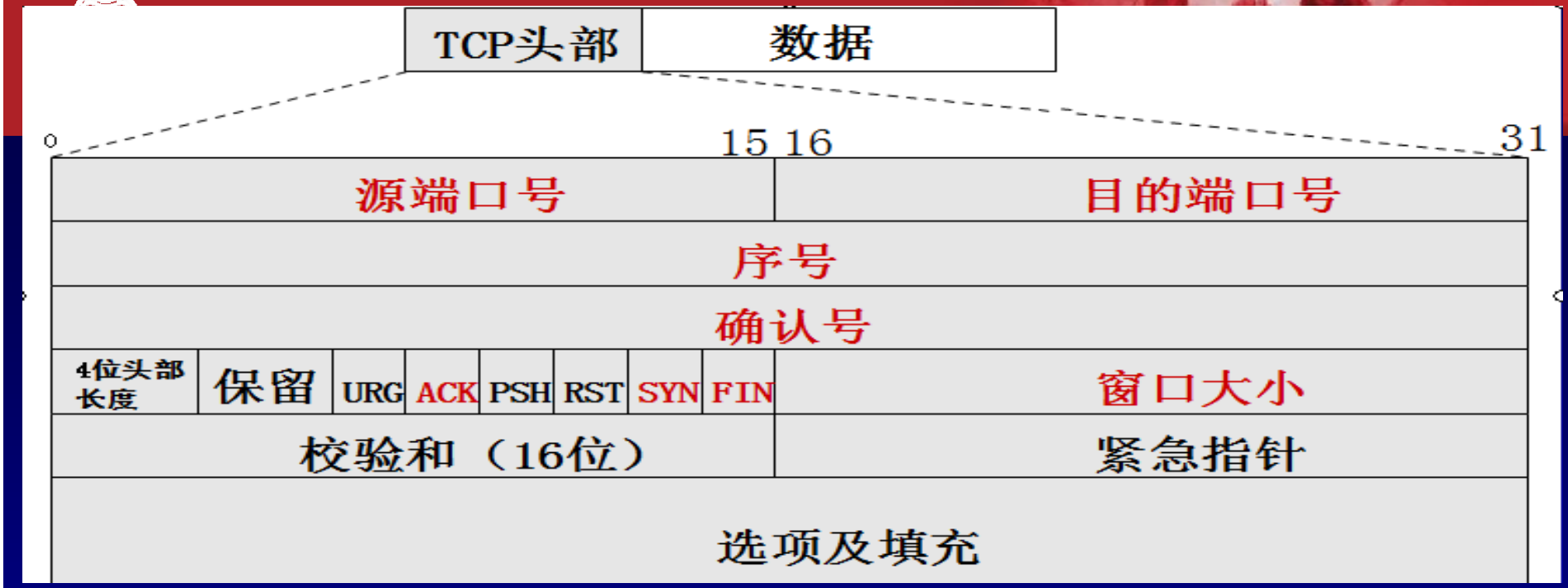


2 序号　占4字节，是本报文段所发送的数据部分第一个字节的序号。在TCP传送的数据流中，每一个字节都有一个序号。例如，在一个报文段中，序号为300，而报文中的数据共100字节。那么在下一个报文段中，其序号就是400。因此TCP是面向数据流的。



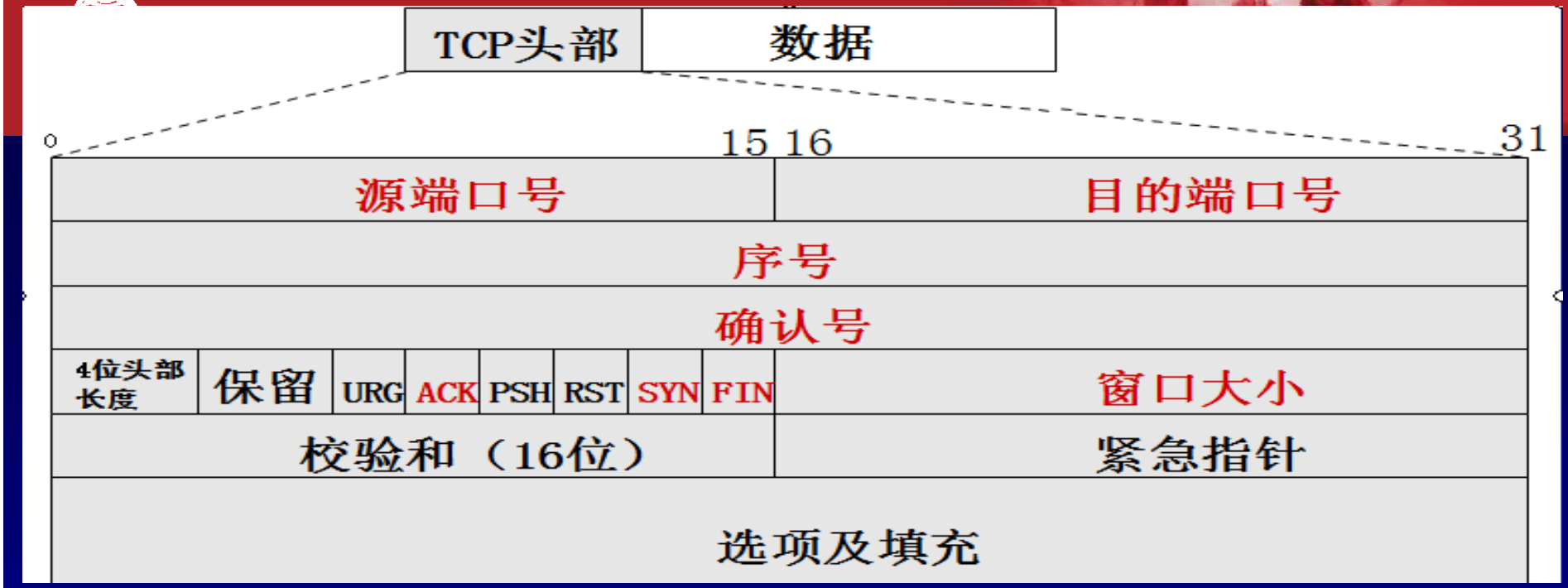
3 确认号 占4字节，是期望收到对方下次发送的数据的第一个字节的序号，也就是期望收到的下一个报文段的首部中的序号。由于序号字段有32bit长，可对4GB(即4千兆字节)的数据进行编号。

4 报头长度 占4bit, 4Byte为单位。

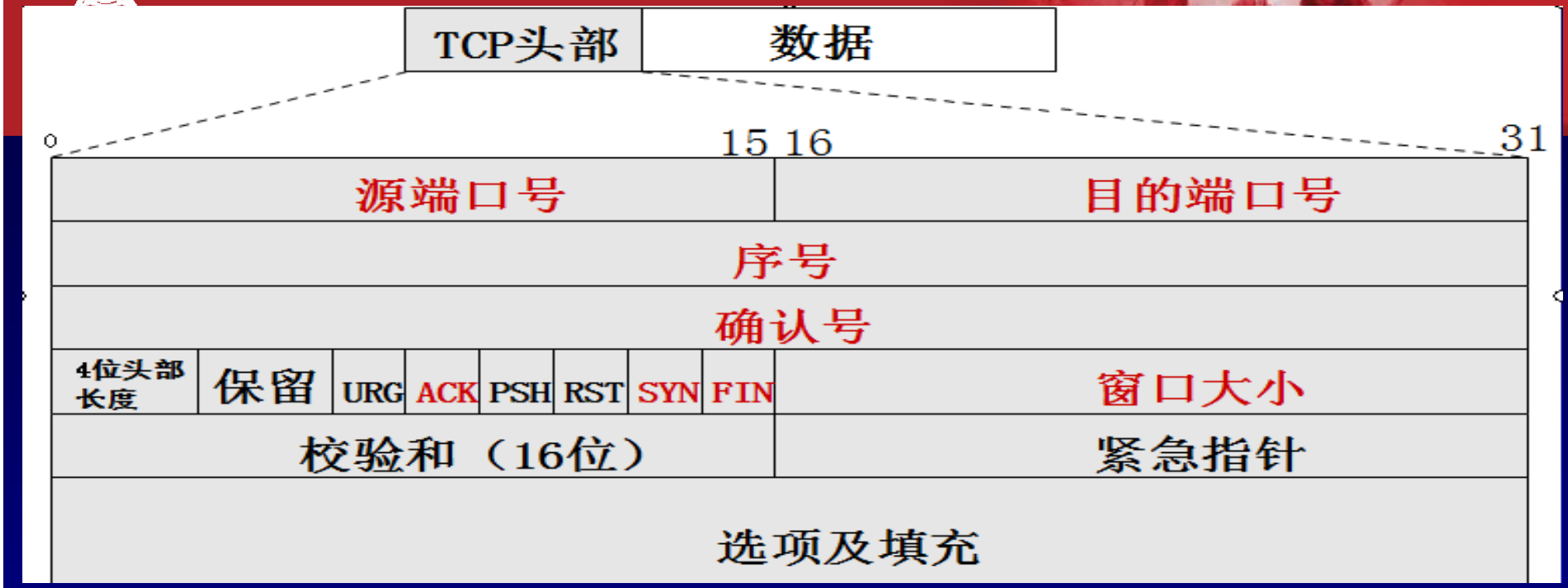


5 确认比特ACK　只有当ACK＝1时确认序号字段才有意义。当ACK＝0时，确认序号没有意义。

6 同步比特SYN　在连接建立时使用。当SYN＝1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，则应在发回的报文段中使SYN＝1和ACK=1。因此，同步比特SYN为1，就表示这是一个连接请求或连接接受报文，而 ACK比特的值用来区分是哪一种报文。



7 终止比特FIN　　用来释放一个连接，当FIN＝1时．表明欲发送的字节串已经发完，并要求释放传输连接。



8 窗口大小　占2字节。窗口字段实际上是报文段发送方的接收窗口，单位为字节。通过此窗口告诉对方，“在未收到我的确认时，你能发送的数据的字节数至多是此窗口的大小。”

9 检验和　占2字节。检验和字段检验的范围包括首部和数据这两部分。

TCP报文可靠传输包含3个步骤

（1）建立连接

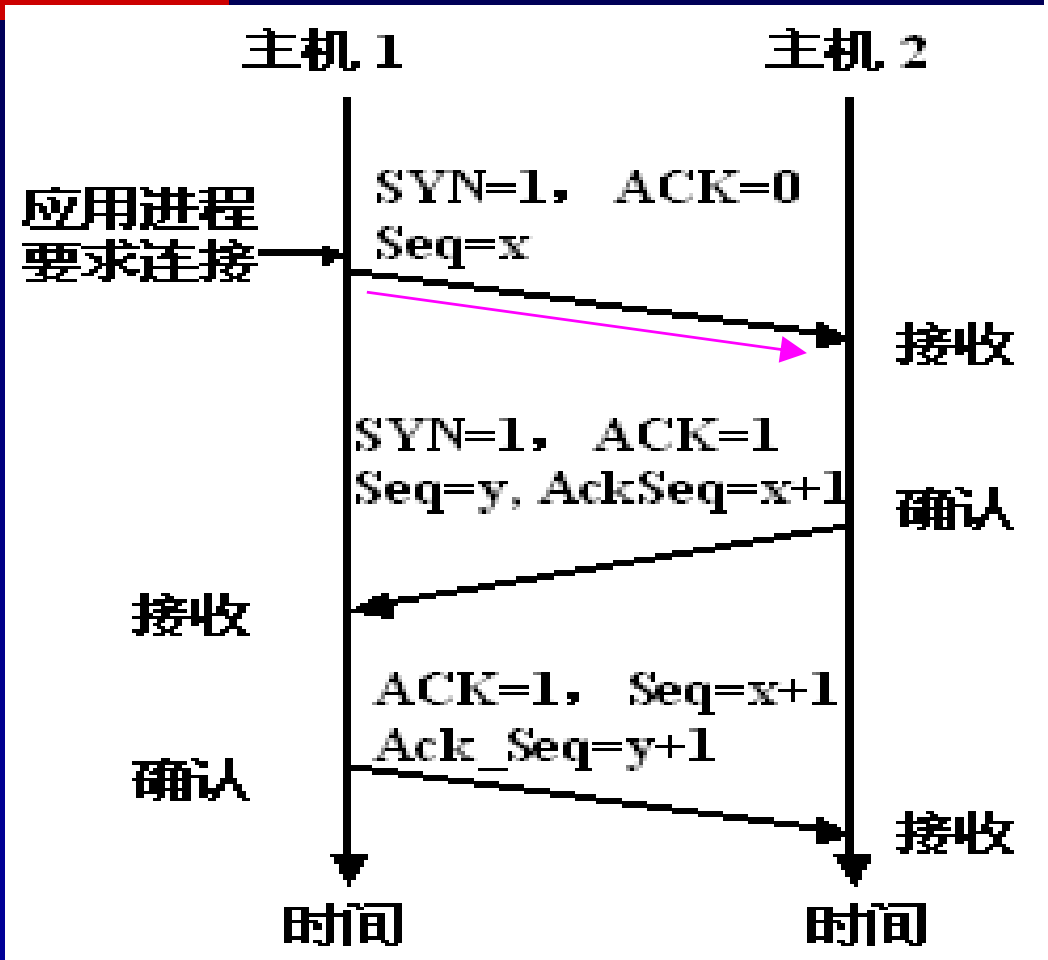
（2）报文传输

- 差错控制：校验，错误重传机制
- 流量控制：滑动窗口机制

（3）传输闭连接

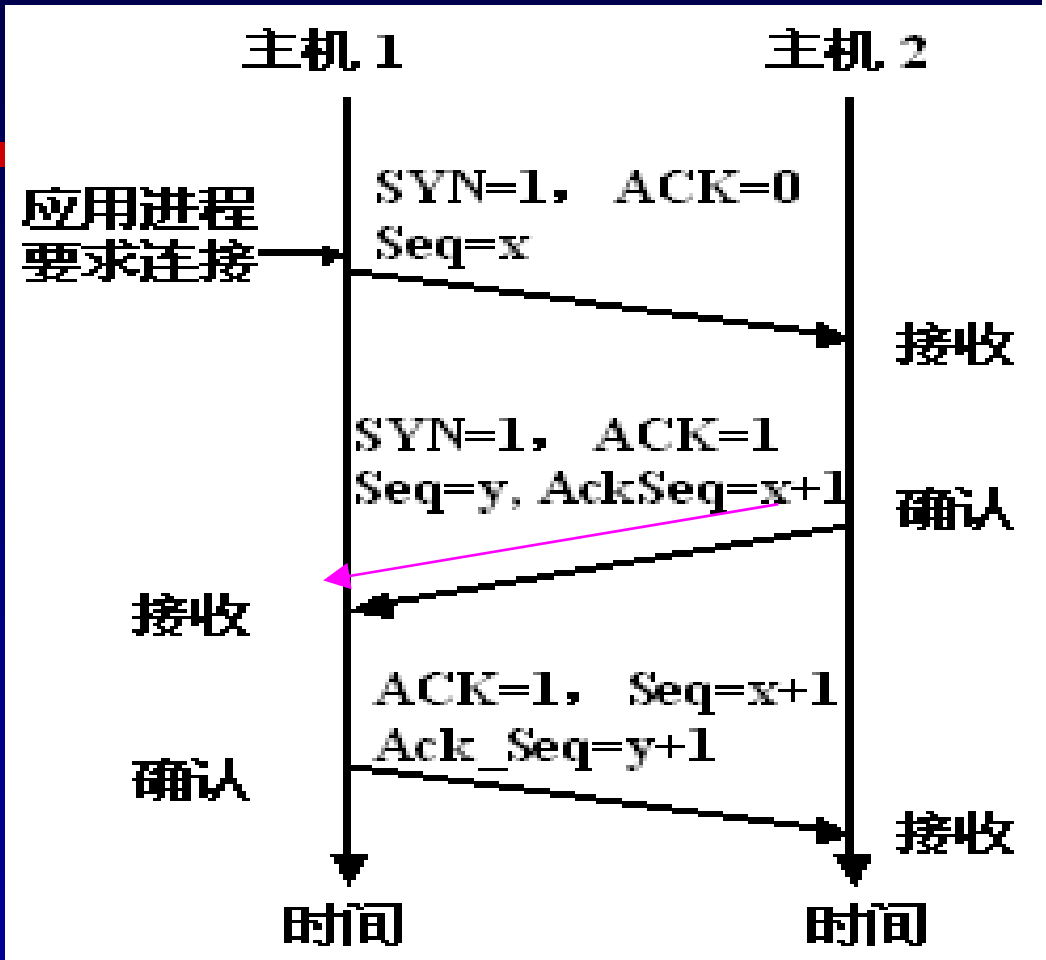
建立TCP连接：三次握手　连接是进程之间的

主机1 首先发起TCP 连接请求，并在所发送的数据段中将控制字段中的SYN置为“1”、ACK置为“0”，并设置数据起始序号为x.



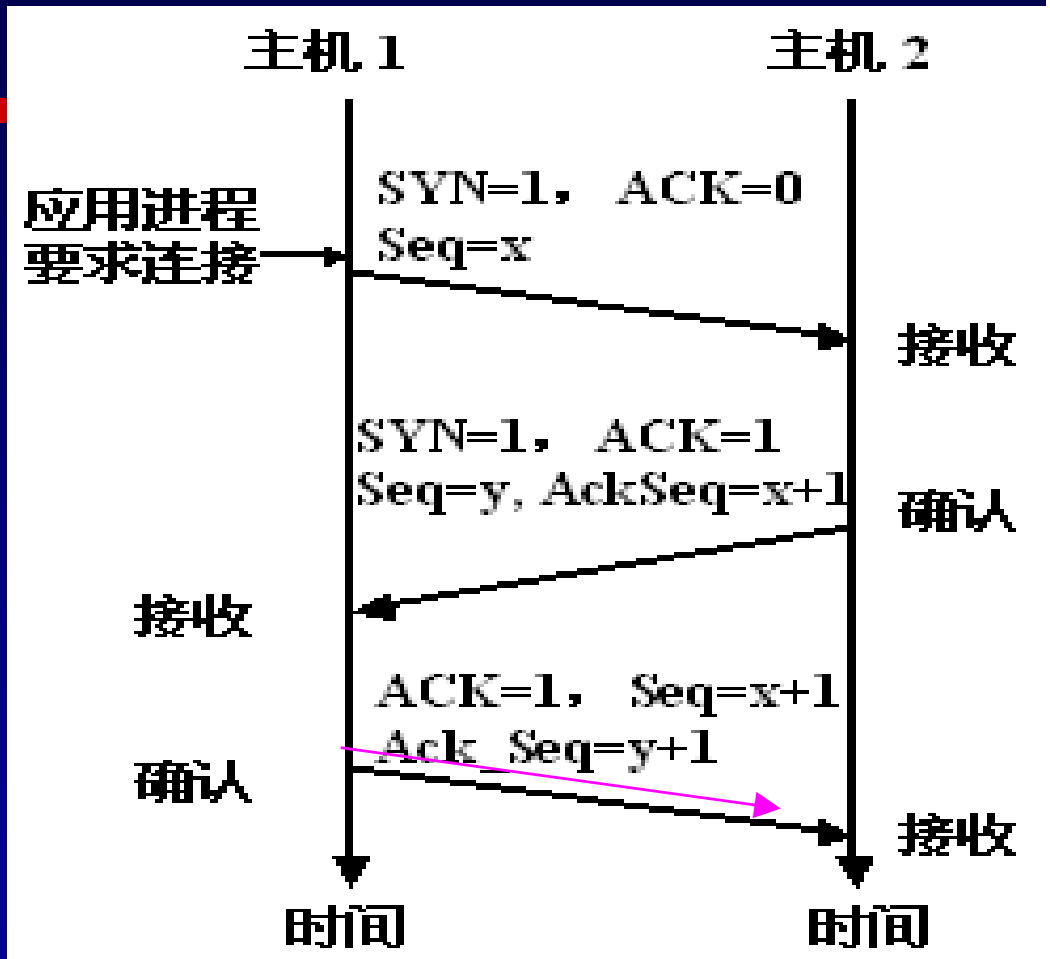
主机2 收到该报文，若同意建立连接，则发送一个连接接受的应答数据段，其中控制字段的SYN 和ACK均被置“1”，指示对第一个SYN 报文段的确认，响应序号x+1,表示对指定报文的响应，同时指定自己的数据序号y，以继续握手操作；

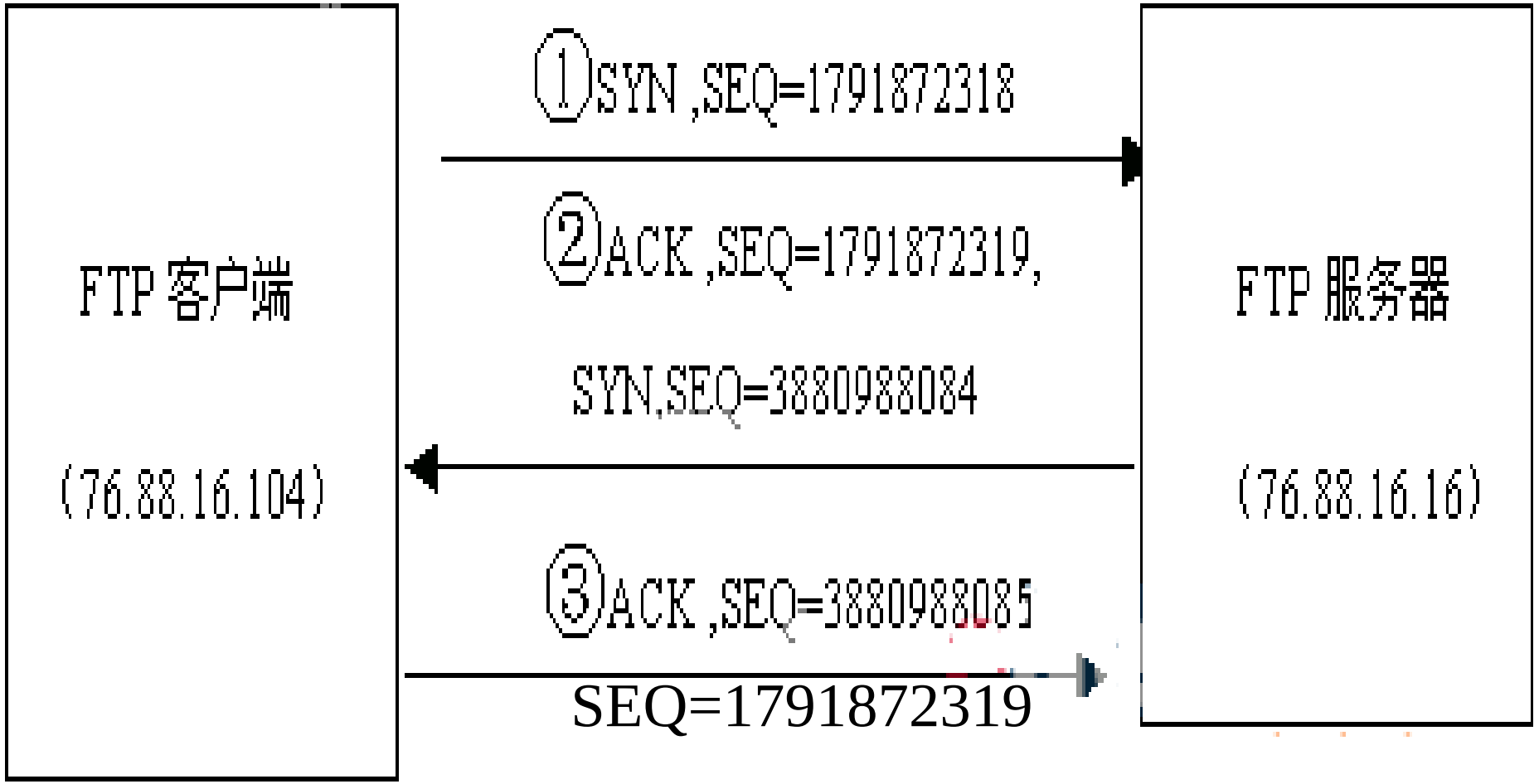
否则，主机2 要发送一个将RST置为“1”的应答数据段，表示拒绝建立连接。



主机1 收到主机2发来的同意建立连接数据段后，还有再次进行选择的机会，若其确认要建立这个连接，则向主机2 发送确认数据段，确认序号为y+1，同时自己的数据需要为x+1，用来通知主机2 双方已完成建立连接；

若主机1已不想建立这个连接，则可以发送一个将RST置为“1”的应答数据段来告之主机2 拒绝建立连接。




FTP 客户端
(76.88.16.104)
①SYN ,SEQ=1791872318
②ACK ,SEQ=1791872319,
SYN,SEQ=3880988084
③ACK ,SEQ=3880988085
SEQ=1791872319
FTP 服务器
(76.88.16.16)

差错控制

TCP采用校验、确认以及超时重传，进行差错控制。

发送方会对发送数据进行处理生成校验码，并设置在校验码字段中，随数据一起发送；

✓接收方对数据进行校验，判断数据传输是否出现错误；

✓接收方对正确接收到的数据进行确认；

✓发送方发送数据时，启动定时器，超时未接收到确认，则重传数据。

TCP的超时重传机制：采用单一定时器

- 发送TCP分段时，如果没有重传定时器开启，那么开启；如果已有重传定时器开启，不再开启。
- 收到一个非重复ACK时，如果有数据在传输中，重新开启重传定时器；如果没有数据在传输中，则关闭重传定时器。
- 未收到确认造成定时器超时，重传所有发出未确认的分段。
- 收到重复ACK时，超过3个，则立即重传重复确认的数据。

TCP采取了延迟确认的机制

- 如果收到的报文段的序号等于rcv_base，并且所有在rvc_base之前的报文段的确认都已经被发送，则只更新rcv_base，但是延迟该报文段的ACK的发送。延迟的ACK可能会在接收端有数据要发送给发送端时被发送或者在接收端有多个ACK需要被发送给发送端时被发送。
- 如果收到的报文段的序号等于rcv_base，并且有延迟的ACK待发送，则更新rcv_base，并发送累积的ACK以确认这两个按序报文段。
- 如果收到的报文段的序号大于rcv_base（乱序），则发送冗余的ACK，即重传对已经确认过的最后一个按序到达的报文段的ACK。接收方会收到冗余ACK。

重传超时时间（RTO，Retransmission TimeOut）

影响超时重传机制协议效率的一个关键参数是重传超时时间（RTO，Retransmission TimeOut）

RTO的值被设置过大过小都会对协议造成不利影响。如果RTO设置过大将会使发送端经过较长时间的等待才能发现报文段丢失，降低了连接数据传输的吞吐量；另一方面，若RTO过小，发送端尽管可以很快地检测出报文段的丢失，但也可能将一些延迟大的报文段误认为是丢失，造成不必要的重传，浪费了网络资源。

传输往返时间（RTT，Round Trip Time）

对一个连接而言，若能够了解端点间的传输往返时间（RTT，Round Trip Time），则可根据RTT来设置一合适的RTO。显然，在任何时刻连接的RTT都是随机的，无法事先预知。TCP通过测量来获得连接当前RTT的一个估计值，并以该RTT估计值为基准来设置当前的RTO。

在发生超时重传时，TCP不是以固定的时间间隔来重传的，而会再每次重传时都将下一次重传的间隔设置为上次重传间隔的2倍，因此重传间隔是倍数增加的。

确认的二义性

如果在一个报文段中的数据被一次性地成功传输和确认，那么发送端可以准确得到该报文段传输的RTT样本。但若出现了重传，情况就会变得很复杂。

例如，一个报文段发送后出现超时，TCP将重传该报文。由于这两个报文段包含了同样的数据，发送方接收到确认信息时将无法分辨出确认信息到底是针对哪个报文段的，因为这两个报文段产生的确认信息可能是完全相同的，确认信息既可能是针对原始报文段的，也可能是对重传报文段的确认。这种现象称为确认二义性。确认的二义性将导致TCP无法准确地估算RTT。

为了避免确认二义性带来的问题，TCP采用了Karn算法来维护RTT的估计值。规定，TCP只能利用没有确认二义性（既无重发、一次发送成功并得到确认的报文段）的RTT样本来对RTT的估计值进行调整。

流量控制

接收方和发送方都有一定大小的接收和发送缓冲区。发送方的发送速度不能超过接收方的接收速度。

为了防止由于发送端与接收端之间的不匹配而引起数据丢失，TCP采用滑动窗口进行流量控制。双方通过窗口大小来告诉对方，在没有收到确认前，最多可以发送的数据量。

流量控制

发送窗口=min{拥塞窗口，通告窗口}

- 拥塞窗口：从通信子网拥塞程度考虑确定的在没有收到接收方确认情况，发送方可以发送的数据量；
- 通告窗口：从接收方接收能力的角度考虑，确定的在没有收到接收方确认情况，发送方可以发送的数据量；由接收到的TCP报文中的窗口大小字段确定。

TCP连接的拆除

TCP连接的拆除通过“四次握手”完成

当主机1 的数据已发送完毕时，其在等待确认的同时可发送一个将控制字段FIN 置“1”的数据段给主机2，表示请求中断主机1到主机2的连接。

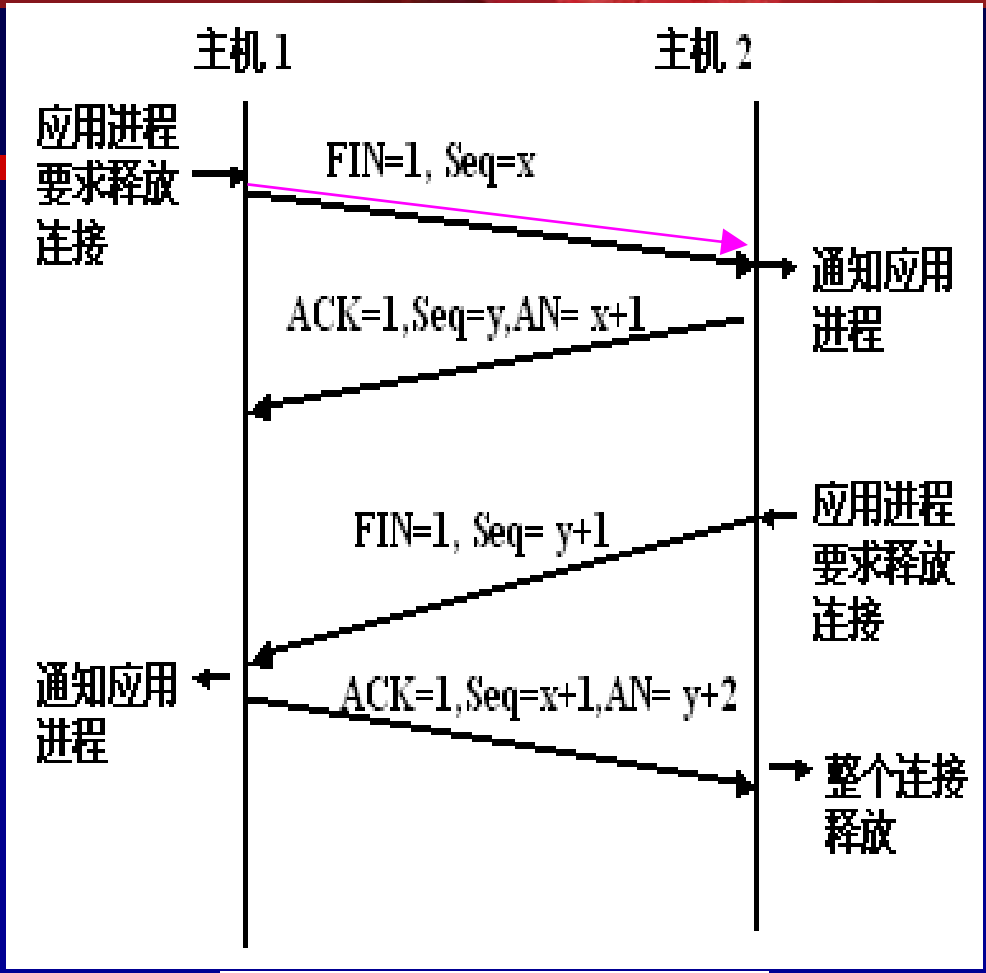


图5-9 四次握手拆除TCP 连接

若主机2 已正确接收主机1 的所有分段，则会发送一个数据段正确接收的确认段，同时通知本地相应的应用程序，对方要求关闭连接，接着再发送一个对主机1 所发送的FIN段进行确认的应答段。由此便拆除了一个方向的TCP连接。

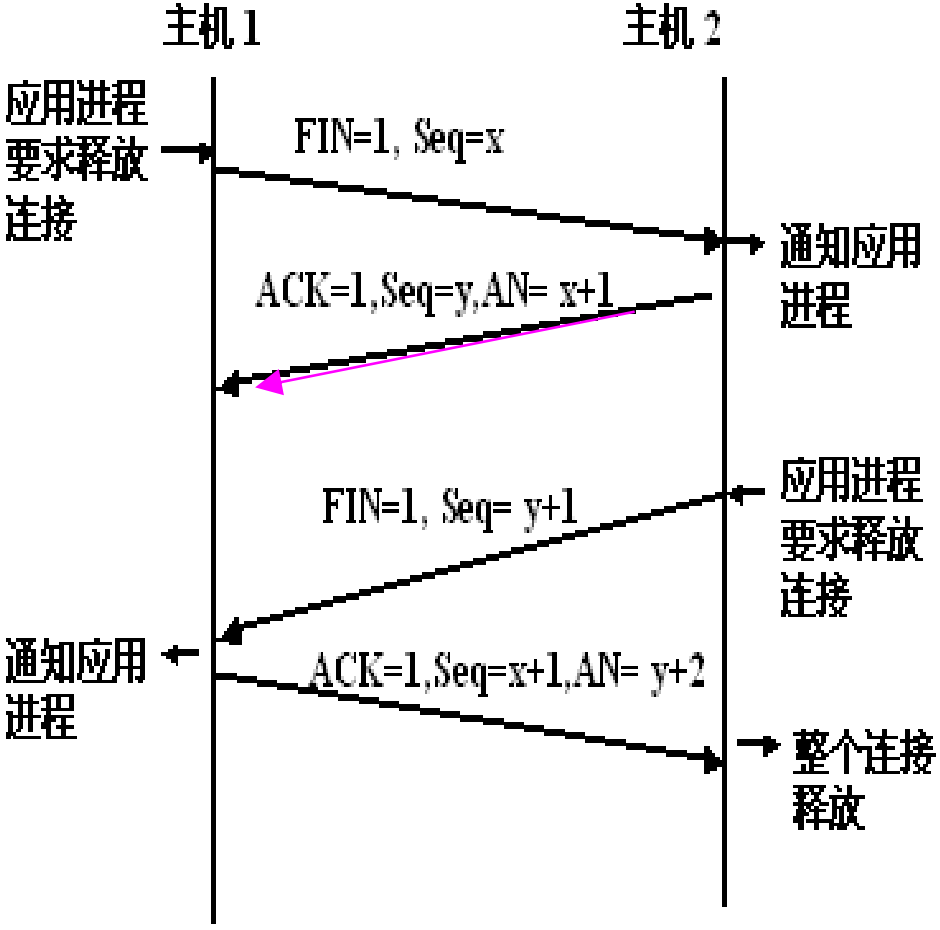


图5-9 四次握手拆除TCP 连接

但是，此时在相反方向上，主机2 仍然可以向主机1 发送数据，直到主机2 数据发送完毕并要求关闭连接。这个方向上连接的拆除同样要经过（1）、（2）两步，由主机2发起FIN段，主机1应答确认ACK，拆除另一方向的TCP连接。

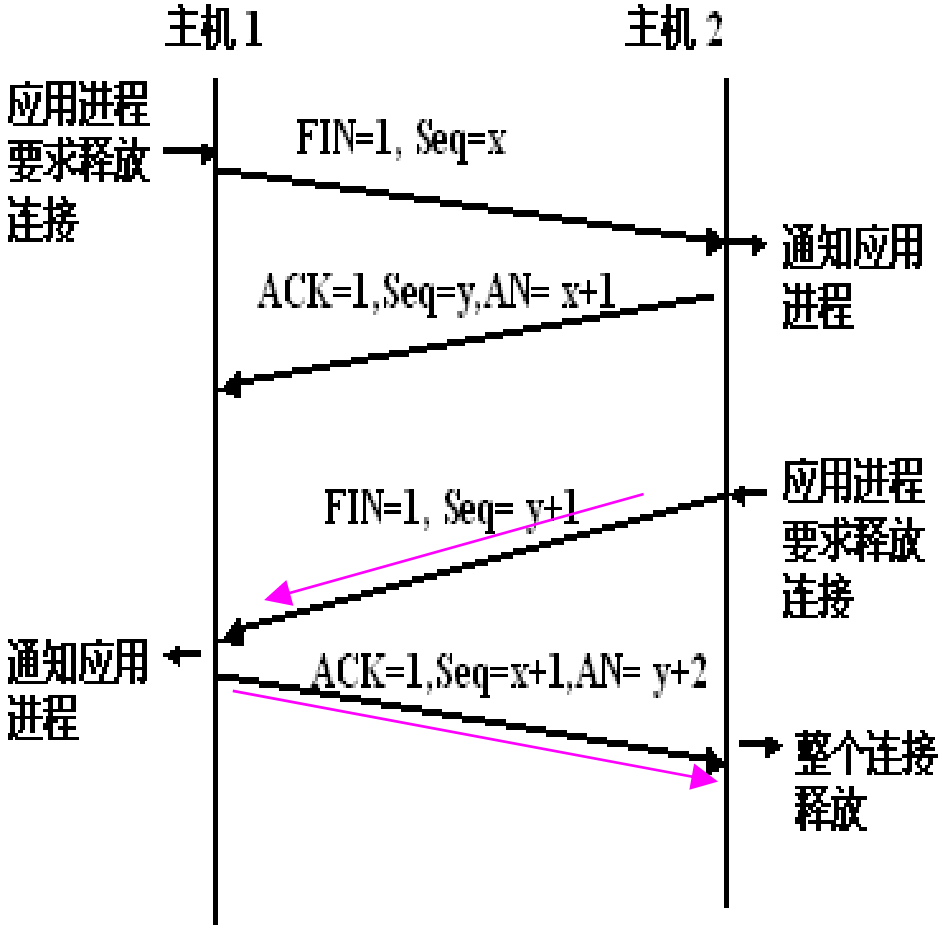


图5-9 四次握手拆除TCP 连接

连接的半关闭

TCP提供了连接的一端在结束它的发送后还能接收来自另一端数据的能力。这就是所谓的半关闭。

常用TCP的端口号分配

端口号	服务进程	说明
20	FTP	文件传输协议（数据连接）
21	FTP	文件传输协议（控制连接）
23	Telnet	虚拟终端网络
25	SMTP	简单邮件传输协议
53	DNS	域名服务器
80	HTTP	超文本传输协议
111	RPC	远程过程调用

TCP连接中的重要定时器

1.重传定时器

2.连接建立定时器

3. ACK延迟定时器

4.持续定时器

5.保活定时器

6.闲置定时器

一、重传定时器（Connection Establishment Timer）

重发定时器是TCP发送数据时设置的，如果在定时器超时前该数据段被确认，就关闭该定时器，否则，一旦超时则重发该数据段。

二、连接建立定时器（Connection Establishment Timer）

当请求建立连接的SYN数据段发出时，连接建立定时器就开始计时，如果在75秒内未收到响应，则连接建立失败。

三、ACK延时定时器（Delayed ACK Timer）

当TCP实体收到数据时它必须返回确认，但并不需要立即回复，它可以在200毫秒内发送ACK报文，如果在这段时间内它恰好有数据要发送，它就可以在数据内包含确认信息，因此需要ACK延时定时器。

四、持续定时器(零窗口探测定时器)

防止死锁事情发生，发送方在收到接收方发来一个窗口为0的数据时，就启动持续定时器，等到该定时器超时还没有收到对方修改窗口大小的数据段时，发送方就发一个探测数据，对接收方对该探测数据的响应应包含了窗口大小，若仍为0，则定时器清0，重复以上步骤，否则则可以发送数据。

当网络中没有发送且未确认的数据包，且本端有待发送的数据包时，启动零窗口探测定时器。

为什么要有这两个限定条件呢？

- 如果网络中有发送且未确认的数据包，那这些包本身就可以作为探测包，对端的**ACK**即将到来。
- 如果没有待发送的数据包，那对端的接收窗口为不为**0**根本不需要考虑。

五、保活定时器（The Keepalive Timer ）

当一个连接长时间闲置会造成保持存活定时器会超时，这时就会发送一个空数据段检测另一方是否仍然存在（即连接是否依然激活），如果它未得到响应，便终止该连接 。

六、闲置定时器（The Quiet Timer）

当TCP连接断开后，为防止该连接上的数据还在网络上，并被后续打开的具有相同五元组的连接接收，要设置闲置定时器以防止刚刚断开连接的端口号被立即重新使用。

TCP的拥塞控制

网络中的链路带宽、交换节点的存储和处理能力等都是网络的资源，这些资源一般是有限的，当网络的资源容量和处理能力大于网络负载的需求时，网络处于正常运转状态，反之网络会出现拥塞。

网络拥塞的根本原因在于端系统向网络提供的负载大于网络资源容量和处理能力，主要体现在网络转发设备的存储空间有限，网络链路带宽有限以及网络转发设备的处理能力有限等。

TCP的拥塞控制

TCP拥塞控制的基本策略是发送端通过跟踪传输数据的丢失现象和往返时延的变化确定网络的传输能力，并以此来调整发送数据率。

TCP的拥塞控制

拥塞窗口是每个TCP端系统在建立连接时创建的拥塞控制量，同样定义为发送端未收到确认时可以连续发送的字节数。拥塞窗口随网络传输能力变化而变化。当网络负载较小时，拥塞窗口可以设置比较大，发之，就要设置成相对较小值。

TCP发送端的发送窗口大小受接收端设置的接收窗口以及TCP拥塞窗口大小的限制，如果这两个窗口不一致时，TCP选择其中较小的一个窗口作为其发送窗口大小。

TCP的拥塞控制

TCP发送端可以通过两种方式检测到发送的数据在网络中丢失，一种是通过超时定时器，超时未收到对发送数据的正确确认，则判定所发数据丢失，另一种方式是，当发送端连续收到多个对其发送的某个数据分组的重复确认时，说明该分组后继分组在传输中出了问题。对两种不同方式检测到的数据包丢弃，TCP发送端采用不同的方式进行拥塞控制。

针对超时重发检测到的数据丢失，TCP发送端采用慢启动和拥塞避免方法，

对通过重复确认发现的数据包丢失，TCP发送端采用快重发和拥塞避免方法进行拥塞控制。

TCP的拥塞控制

- 两个控制变量
  - 拥塞窗口cwnd
  - 慢启动门限ssthresh
- 两个算法
  - 慢启动算法
  - 拥塞避免算法

拥塞窗口cwnd

- 拥塞窗口cwnd是拥塞控制中TCP能发送的字节数
- TCP能发送的字节还受到流量控制中对方TCP通告窗口的限制
- 因此，TCP可以发送的字节数=min{拥塞窗口，通告窗口}
- 拥塞窗口是发送方感受到的网络拥塞的估计，通告窗口则与接收方在该连接上的可用缓存大小有关

慢启动门限ssthresh

- ssthresh是一个动态变化的门限值，用来确定当前采用慢启动还是拥塞避免算法
  - 拥塞窗口cwnd大于ssthresh时，采用拥塞避免算法
  - 拥塞窗口cwnd小于等于ssthresh时，采用慢启动算法

慢启动算法

- (1) 刚建立连接，cwnd=1 MSS（最大报文段长度）
- (2) 在RTT内收到每收到一个ACK，cwnd增加一个MSS。这样，每个RTT时间内，cwnd会翻倍，cwnd以指数形式增长。
- (3) cwnd会以指数形式增长，直到cwnd达到了门限值ssthresh或发生丢包

ssthresd初始值为65535.

拥塞避免算法

- 当拥塞窗口cwnd的值超过了门限值ssthresh，就进入拥塞避免阶段。此时每个RTT时间内，cwnd的值最多增加一个MSS，此时cwnd是线性增长

发生超时，进入慢启动；超过门限，进入拥塞避免

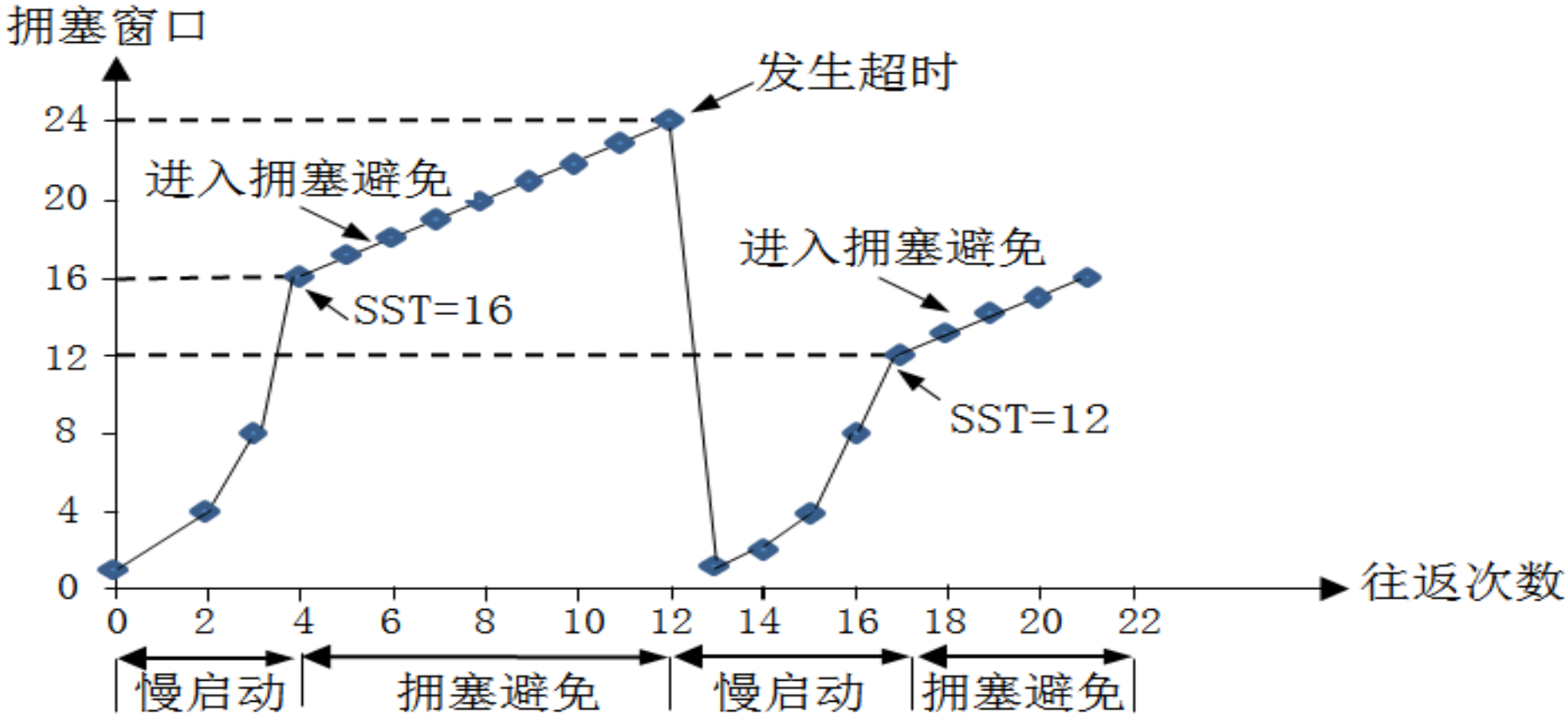
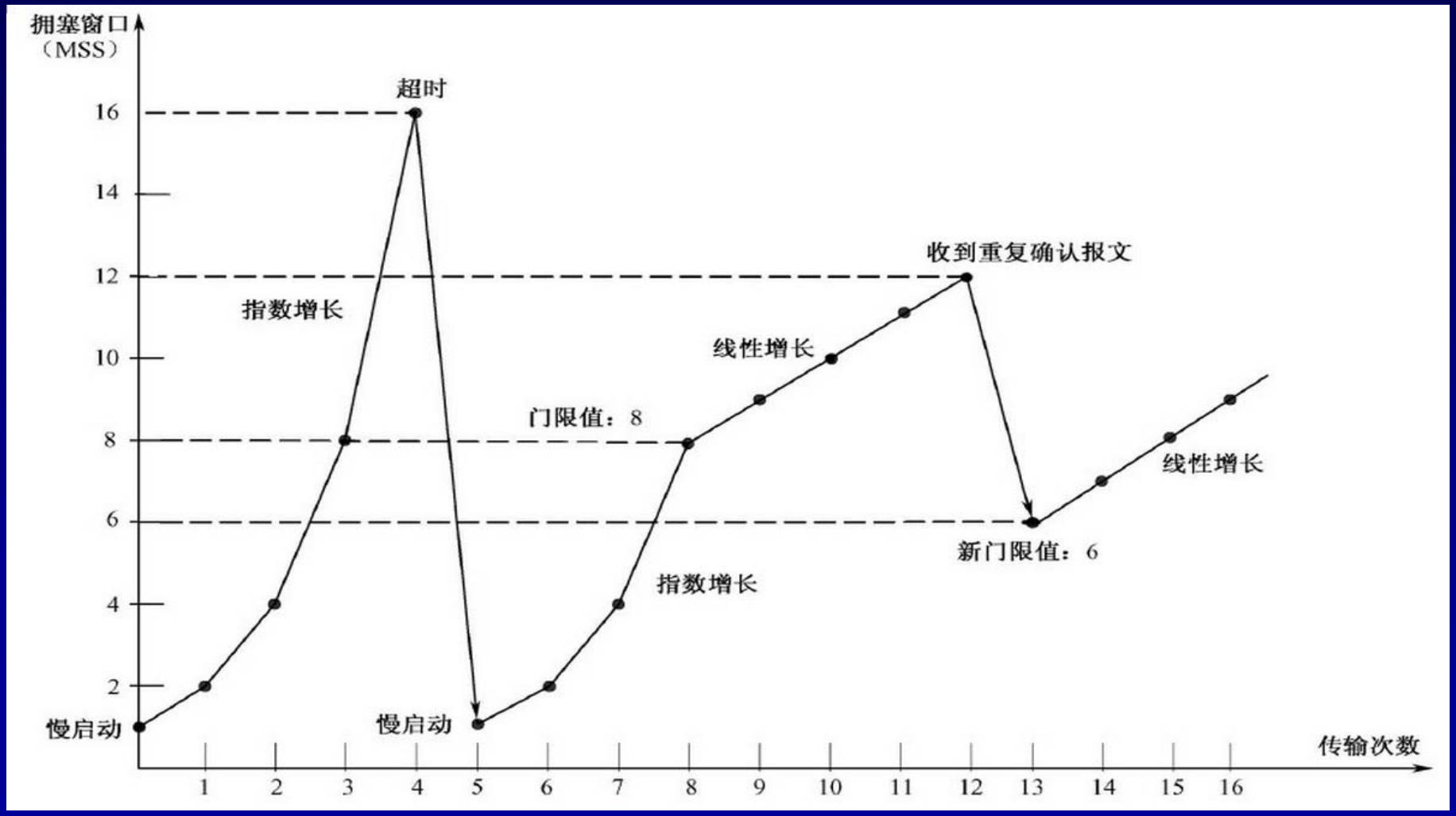


图 7-13 慢启动和拥塞避免

拥塞控制工作过程

- (1)初始化*cwnd*为1个MSS, *ssthresh*为65535字节
- (2)当新的数据被对方确认时，就增加*cwnd*
  - 如果*cwnd<=ssthresh*，就进行慢启动，cwnd指数增长
  - 否则进行拥塞避免，cwnd线性增长
- (3)拥塞发生时，ssthresh设置为cwnd的一半
  - 如果超时，cwnd设为1 MSS，进入慢启动
  - 如果收到重复确认，cwnd设置为拥塞时cwnd值的一半，进入拥塞避免

拥塞控制样例


拥塞窗口
（MSS）
超时
16
14
12
收到重复确认报文
指数增长
10
线性增长
门限值：8
8
线性增长
6
新门限值：6
指数增长
4
2
慢启动
慢启动
传输次数
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

作业

1. 简述TCP建立连接的三次握手过程。
2. 简述TCP拆除连接的四次握手过程。
3. 简述TCP流量控制的基本原理。
4. 什么是拥塞控制，有什么作用？
5. 什么是慢启动、快重传和拥塞控制？