

计算机网络原理与实践（第2版）配套课件
机械工业出版社 2013年

第6章 传输层



本章内容

- 6.1 传输层的基本概念
- 6.2 因特网上的用户数据报协议
- 6.3 因特网上的传输控制协议
- 6.4 用于多媒体传输控制的实时传输/传输控制协议



- 网络层已经保证了寻址到目的主机，那么传输层的任务是什么？
- 传输层的端到端通信是什么含义，是指源端点到目的端点之间的通信，还是指端点应用进程之间的通信？
- 传输层向上层提供哪两种不同的服务？



6.1 传输层的基本概念

- 面向连接和无连接服务
- 因特网上的端到端通信
- 端口和套接字的概念
- 传输层的多路复用与多路分解




6.1.1 面向连接和无连接服务


- 计算机网络通常提供两种类型的服务
 - ✓ 面向连接服务（connection-oriented service）
 - ✓ 无连接服务（connectionless service）



1. 面向连接服务

- 数据传输之前发送方必须与数据接收方建立连接，然后才可以进行数据交换。数据传输结束后终止连接，以释放系统资源。
 - 包括连接建立，数据传输和连接释放三个阶段。
 - 按序传送，可靠性高。适合于在一定期间内向同一数据接收方发送大量报文的情况。
 - 传输层面面向连接的协议：TCP。
- 

2. 无连接服务

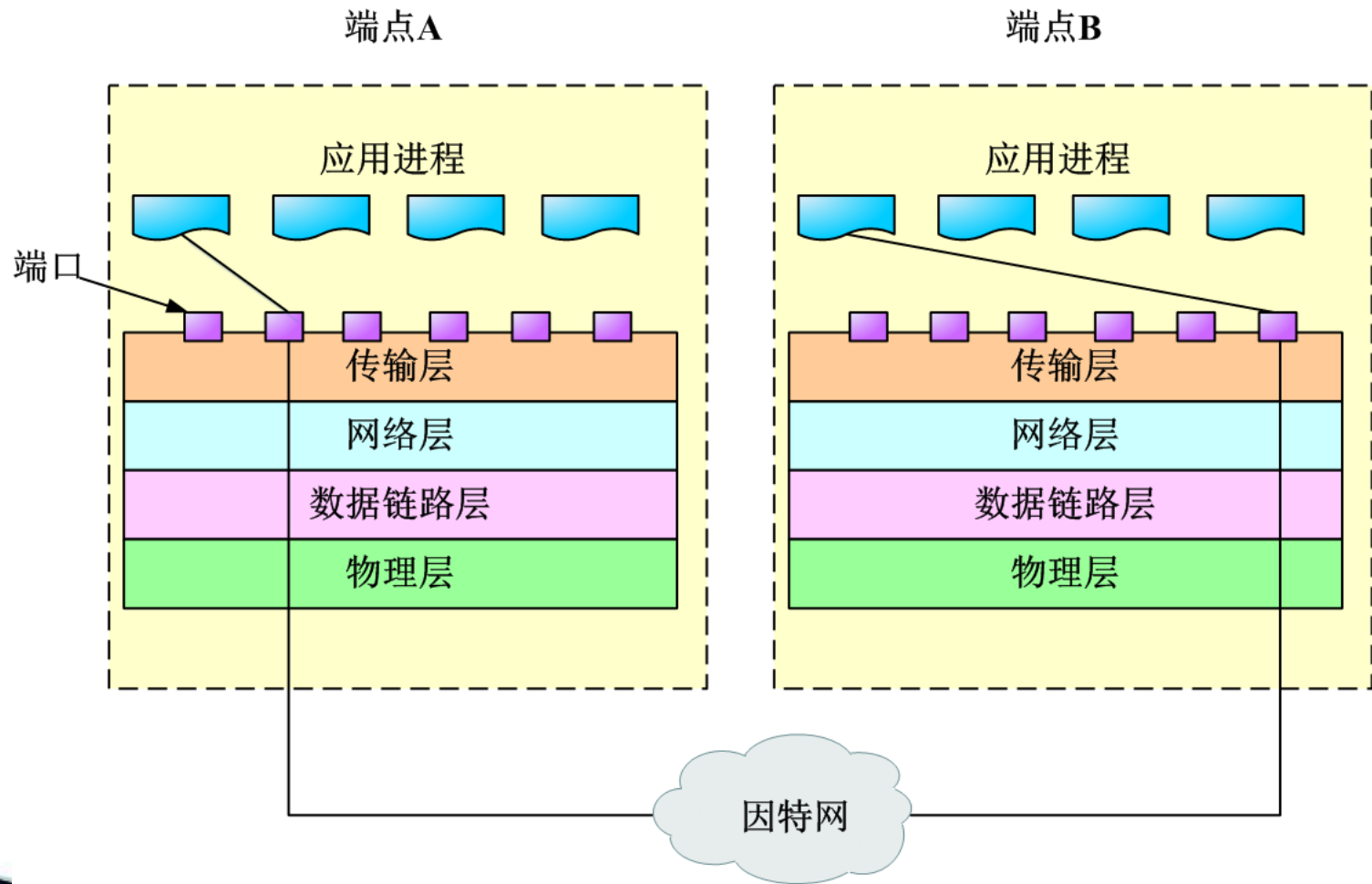
- 进行数据传输前发送方不必与数据接收方建立连接，可直接向接收方发送数据。
 - 无连接服务的特点：
 - ✓ 使用方便灵活
 - ✓ 开销小
 - ✓ 通信迅速
 - ✓ 可靠性低
- 

6.1.2 因特网上的端到端通信

- 端到端的通信是指端点应用程序进程之间的通信。
- 传输层协议的主要任务是保障端点应用程序进程之间的通信，因此传输层协议也被称为端到端协议（end-to-end protocol）。



端到端应用程序进程之间的通信



6.1.3 端口和套接字的概念

- 传输层通过端口控制机制实现端点应用程序进程之间的通信。
- 传输层的端口并非是物理意义上的端口，而是逻辑意义上的端口。
- 端口通过端口号来标记，范围为0~65535。



传输层的端口号分类

1) 熟知端口 (well known ports)

该类端口号范围为0~1023，它们通常绑定于一些众所周知的服务，这些端口由系统调用。



部分常用的熟知端口号

端口号	应用协议
21/20	文件传输协议
23	远程终端协议
25	简单邮件传输协议
53	DNS
67/68	动态主机配置协议
80	超文本传输协议
161	简单网络管理协议

2) 注册端口 (registered ports)

该类端口号范围为1024~49151。它们通常用于众所周知服务之外的不常用的服务。

3) 动态和/或私有端口 (dynamic and/or private ports)

该类端口号范围为49152~65535，该类端口不可以被正式地注册占用。

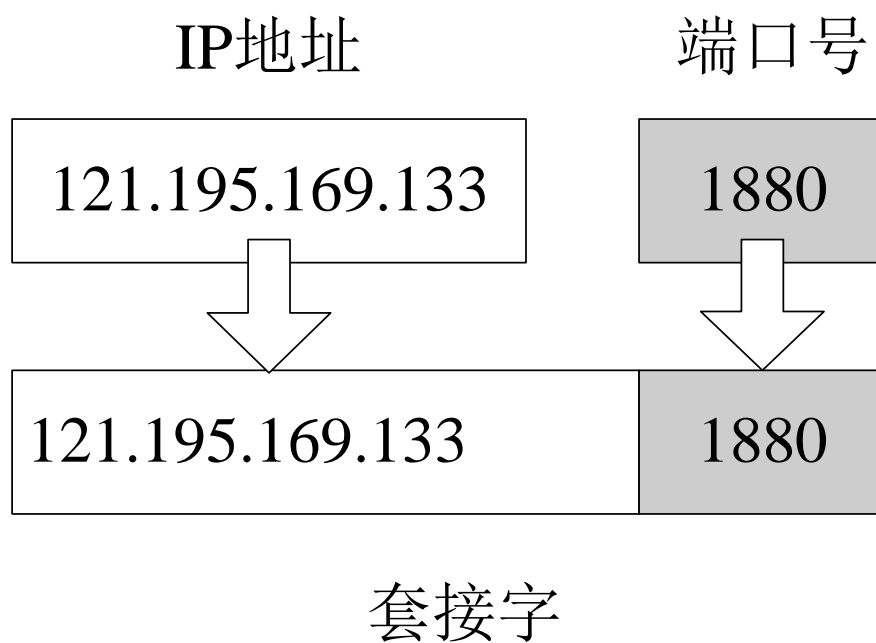


套接字（socket）

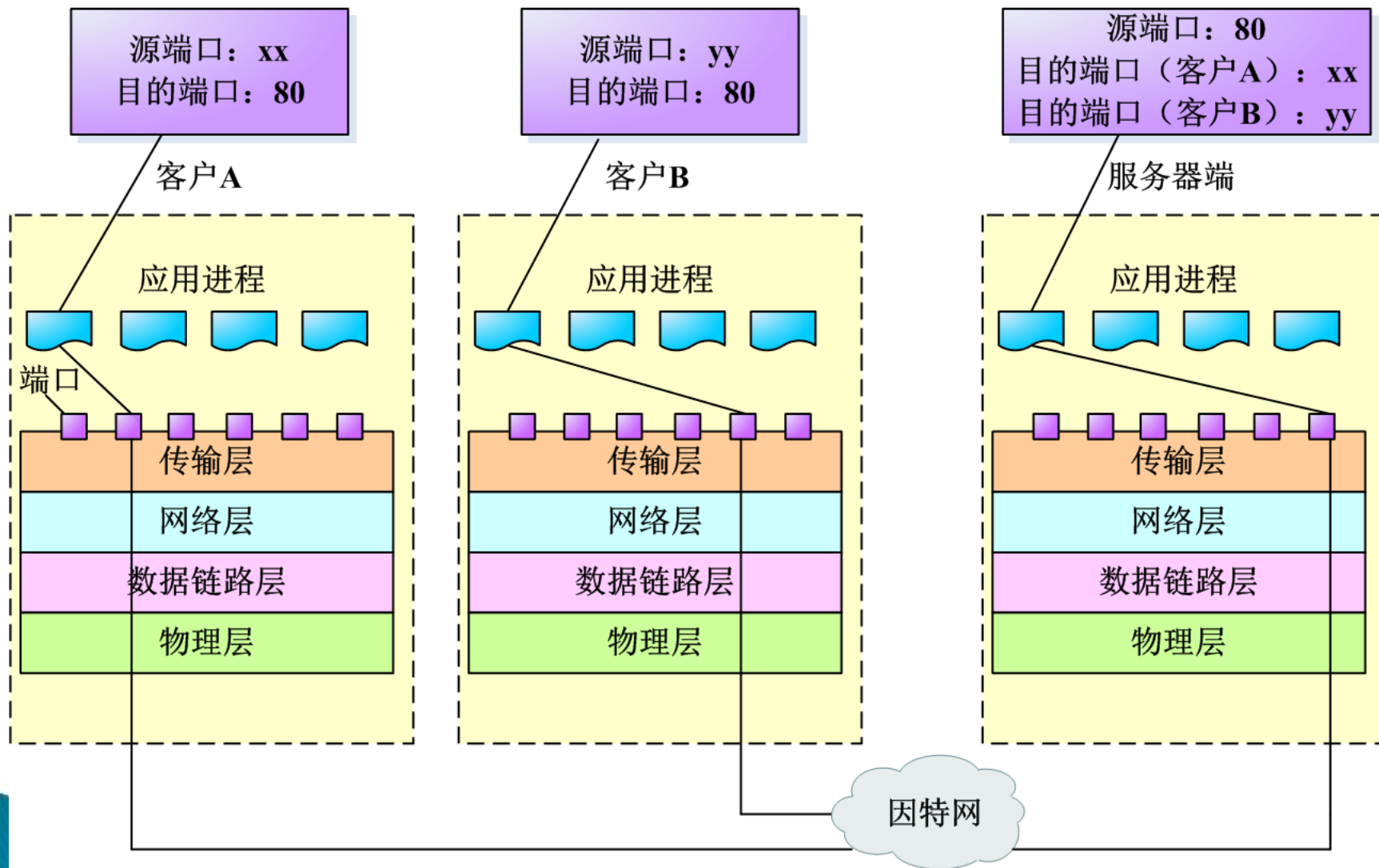
- 为了区分不同的网络应用服务，就必须把主机的**IP**地址和端口号进行绑定后使用。
- 主机**IP**地址和端口号的绑定组成了套接字（socket）。




主机IP地址、端口号和套接字的对应关系



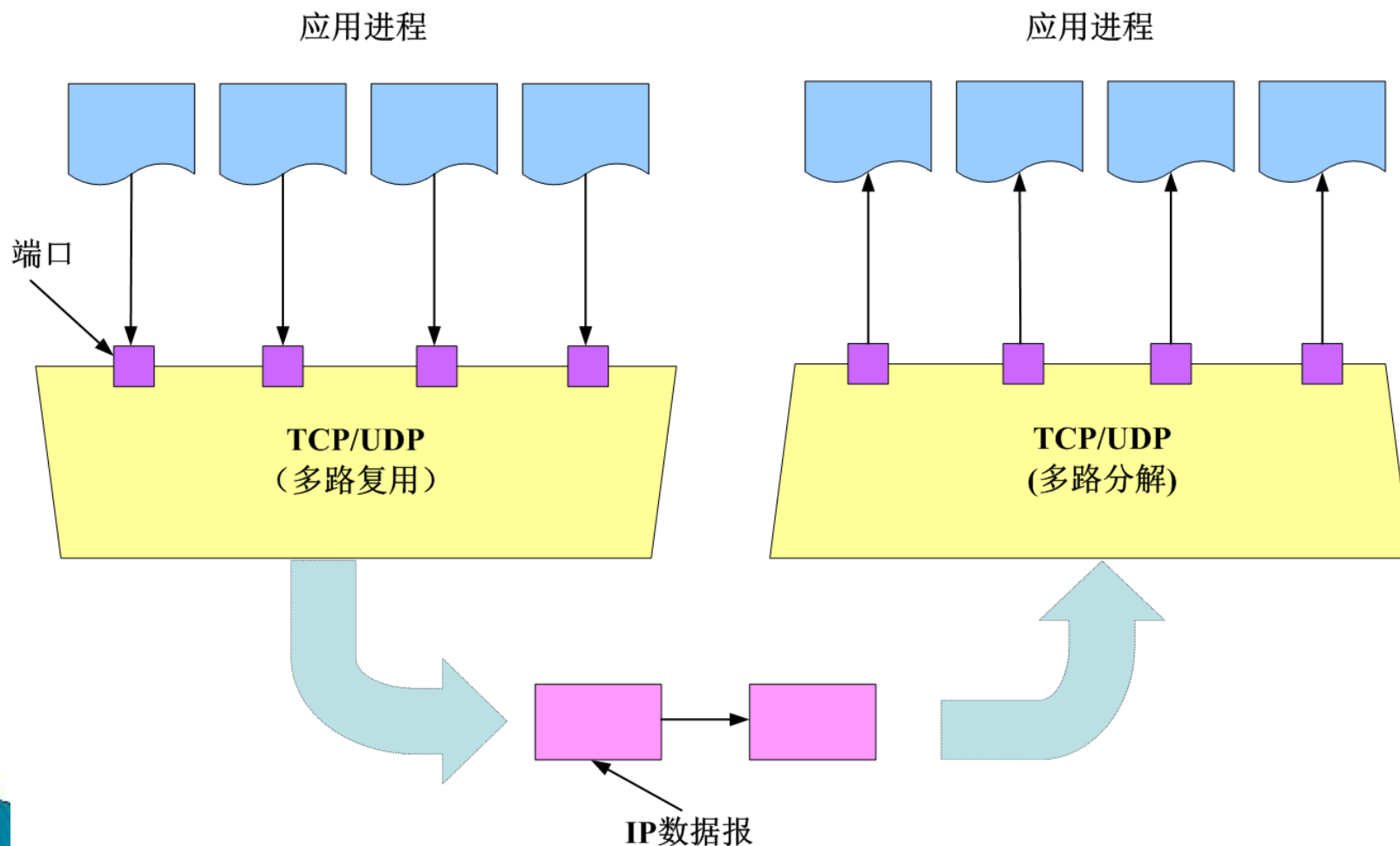
端到端多应用程序进程之间的通信



6.1.4 传输层的多路复用 与多路分解

- 传输层端到端通信提供对应用报文的多路复用和多路分解功能（在传输层上）。
 - 传输层的多路复用：多个应用程序进程使用同一个传输层协议发送数据报。
 - 传输层的多路分解：传输层协议将接收到的报文分发给不同的应用程序进程。
 - 多路复用和多路分解功能通过传输层协议的端口机制实现。
- 

传输层的多路复用和多路分解




6.2 因特网上的用户数据报协议

- UDP概述
- UDP数据报结构
- UDP校验和



6.2.1 UDP概述

- 用户数据报协议（User Datagram Protocol, UDP）是向上层提供数据报服务的简单的无连接传输层协议。
 - 它在网络层IP的基础上，增加了端口机制和简单的校验机制。
 - UDP为因特网应用提供最基本和最简便的通信服务。
 - RFC 768对UDP做出了明确定义和规范。
- 

采用UDP的因特网应用

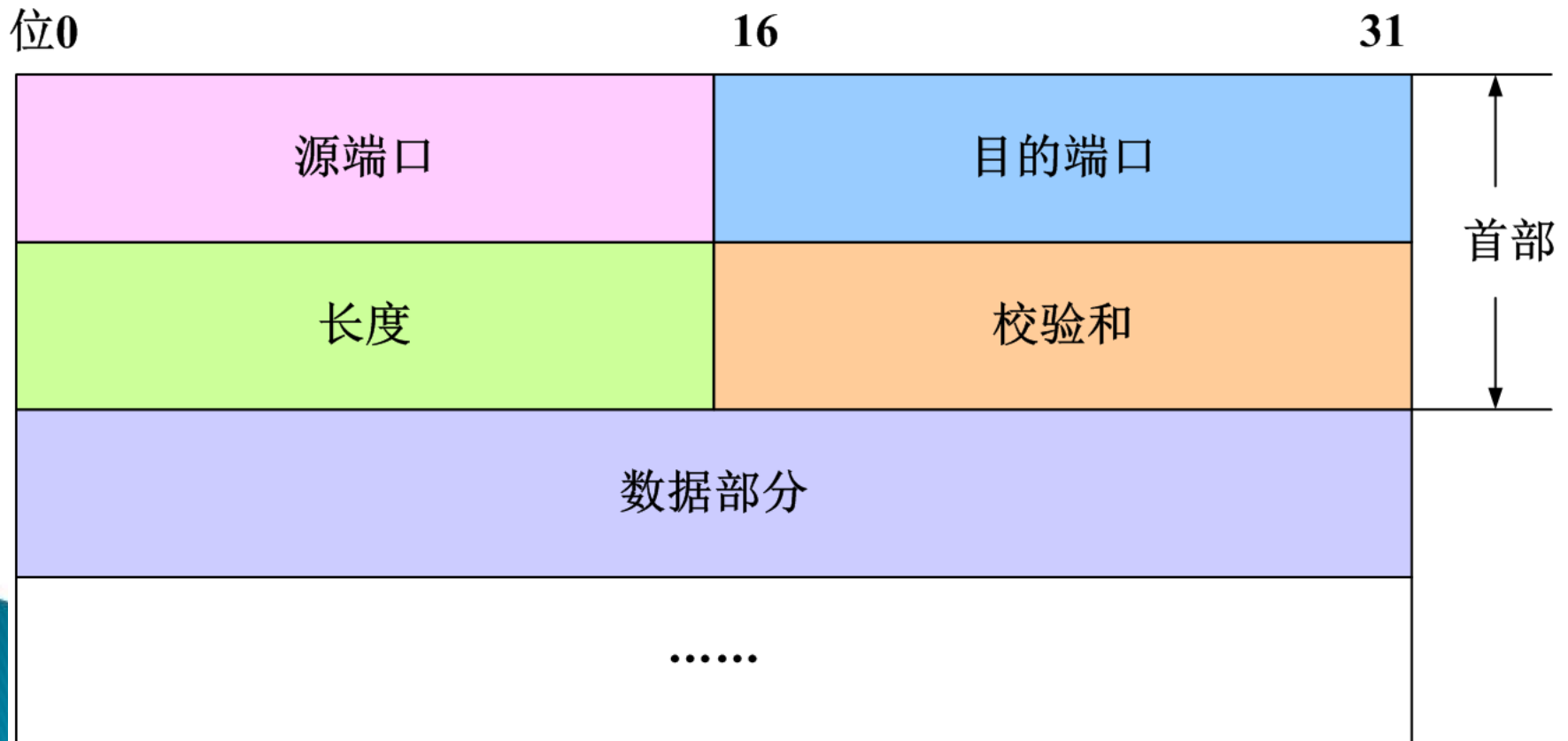
应用	应用层协议	运输层协议
远程文件服务器	NFS	UDP
网络管理	SNMP	UDP
IP电话	专用	UDP
流式多媒体	专用	UDP
域名服务	DNS	UDP

6.2.1 UDP概述

- UDP具有如下特点：
 - ✓ 无连接
 - ✓ 尽力而为的传输
 - ✓ 首部开销小
 - ✓ 无拥塞控制

6.2.2 UDP数据报

- UDP数据报由两部分组成：
UDP数据报首部和UDP数据报数据部分。




- 源端口是可选字段，长度为2个字节。
 - 被使用时，它指向源应用程序进程端口；
 - 若源端口不被使用，可将该字段填充为0。
- 目的端口长度为2个字节，只有在拥有特定的目的网络地址时才有意义。



- 报文长度字段：长度为2个字节，报文总长度，是UDP数据报首部和UDP数据报数据部分的总的字节数，其长度单位为字节。
- 校验和：可选字段，长度为2个字节。主要用来检测UDP报文是否出错。当该字段不被使用时，可将其值设置为0。




UDP伪首部

- 计算校验和时，需要对UDP首部进行扩充。
 - 扩充部分称为UDP伪首部，长度为12字节。
 - 伪首部信息来源于IP首部部分字段：
 - 源IP地址，目的IP地址
 - 0填充字段（使扩充部分长度为4B的整数倍）
 - 所用协议（UDP为17）
 - UDP长度
 - 伪首部只在计算校验和时使用。
- 

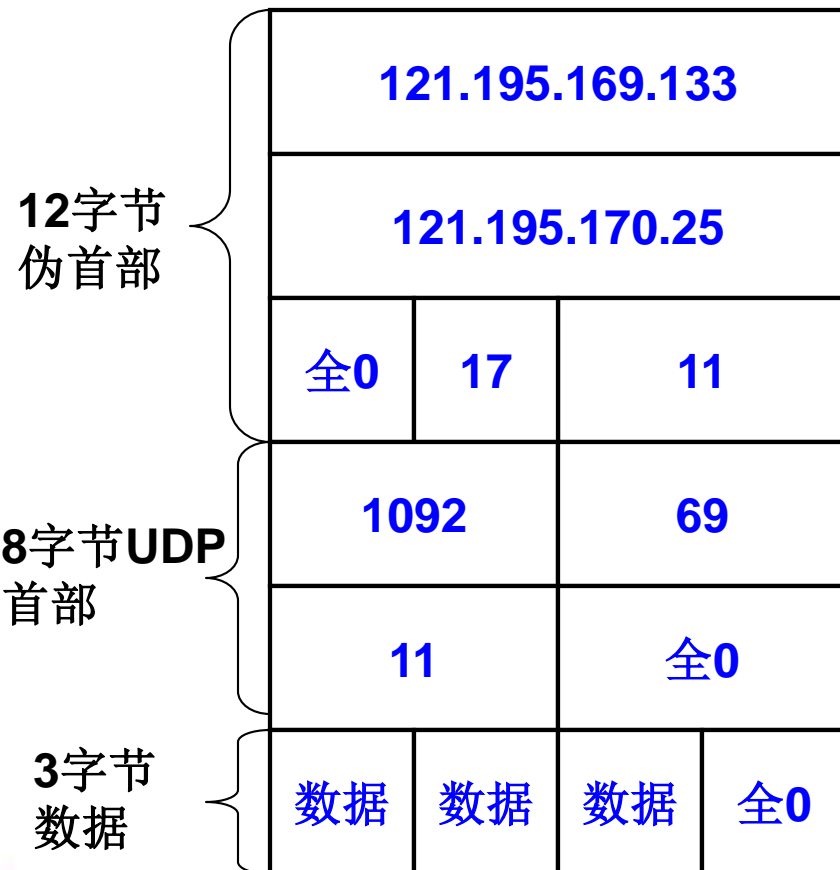
UDP数据报伪首部结构



6.2.3 UDP校验和

- UDP通常采用检验和法对数据报进行检测。
 - 源端口采用二进制反码求和运算计算校验和
 - 先将校验和字段清零
 - 每两字节为一个操作数相加
 - 运算中最高位出现进位时回卷与结果相加
 - 最后结果取反成为校验和字段的值
 - 目的端口对收到的数据报进行同样的运算，求和结果为全1时报文无差错。
- 

计算UDP校验和示例



填充

求反码

01111001	11000011	→	121.195
10101001	10000101	→	169.133
01111001	11000011	→	121.195
10101010	00011001	→	170.25
00000000	00010001	→	0和17
00000000	00001011	→	11
00000100	01000100	→	1092
00000000	00001011	→	11
00000000	00000000	→	0(校验和)
00110101	01000111	→	数据
01010001	00000000	→	数据和0


11010010 00011011 → 求和得出的结果

00101101 11100100 → 校验和

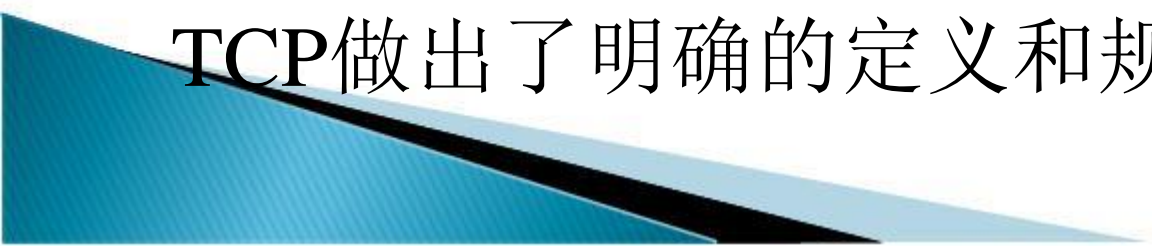
RUDP

- UDP提供不可靠的数据报服务
- 可以考虑通过并在应用服务中自行增加报文序号、报文确认、报文重传、连接控制、流量控制和拥塞控制等可靠性机制，实现使用UDP进行可靠的端到端的通信。
- 这种改进的UDP协议通常称为可靠的UDP（Reliable UDP，RUDP）。

6.3 因特网上的传输控制协议

- TCP概述
 - TCP报文段结构
 - TCP序号与确认
 - TCP重传机制
 - TCP连接管理
 - TCP流量控制
 - TCP拥塞控制
- 

6.3.1 TCP概述

- 传输控制协议（Transmission Control Protocol, TCP）是面向连接、面向字节流的可靠的传输层协议。
 - TCP采用报文序号、报文确认、报文重传、连接控制、流量控制和拥塞控制等一系列可靠性机制，为因特网应用提供可靠的通信服务。
 - IETF RFC 793、RFC 1323和RFC 2581等对TCP做出了明确的定义和规范。
- 

采用TCP的因特网应用

应用	应用层协议	运输层协议
电子邮件	SMTP	TCP
远程终端访问	Telnet	TCP
万维网	HTTP	TCP
文件传输	FTP	TCP

- TCP具有如下特点：
 - ✓ 面向连接
 - ✓ 面向字节流
 - ✓ 全双工通信
 - ✓ 提供可靠的传输服务

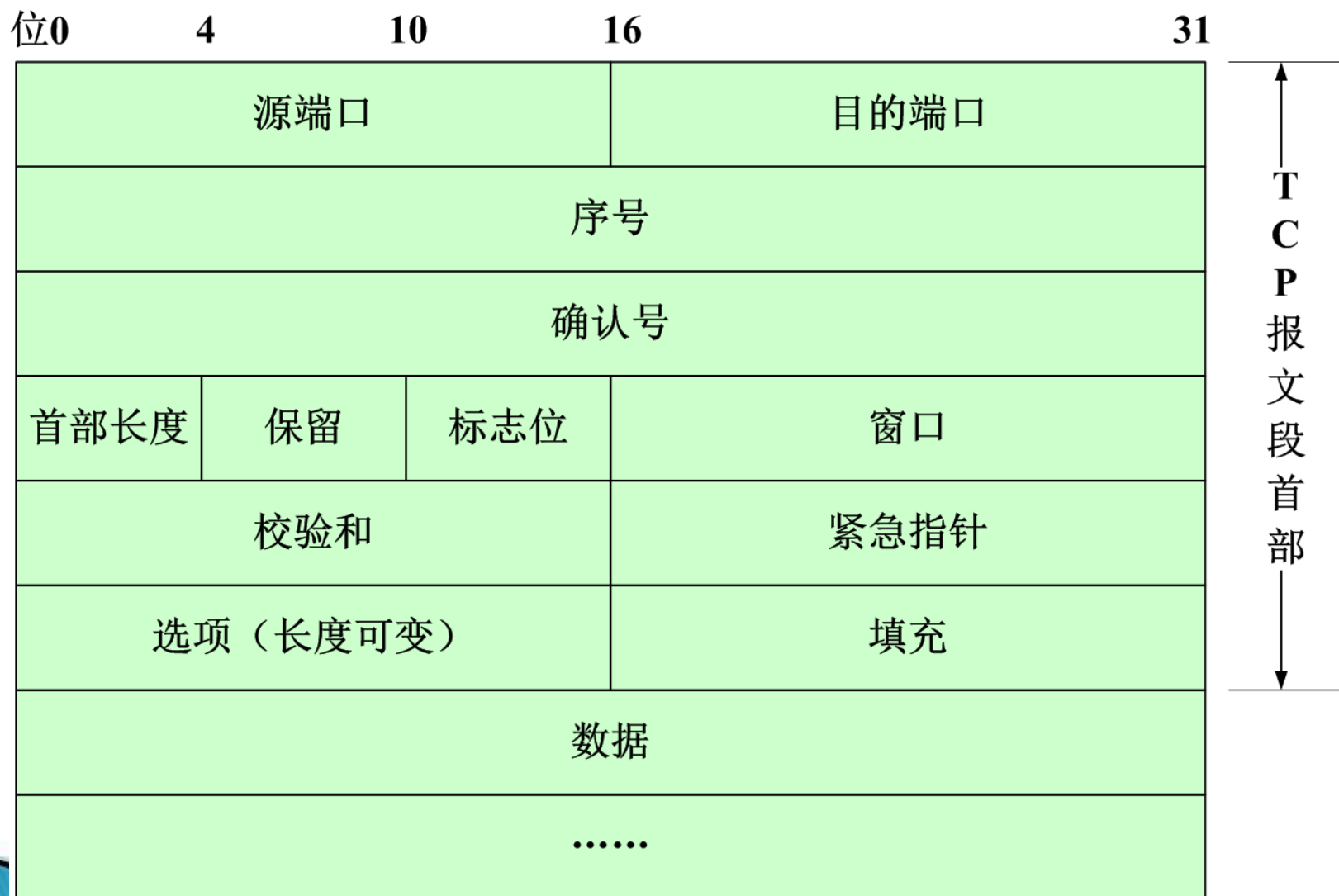


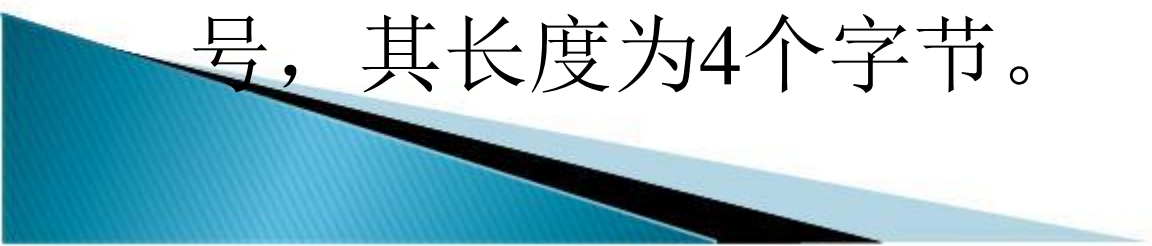
6.3.2 TCP报文段结构


- TCP是面向字节流的数据通信协议，我们将源端口和目的端口之间传输的数据单元称为TCP报文段。
- TCP报文段的组成：
 - ✓ TCP报文段首部
 - ✓ TCP报文段数据部分



TCP报文段结构



- 源端口和目的端口：分别用来标识发送和接收TCP报文段的应用程序进程，与IP数据报中的源IP地址和目的IP地址组成插口，以唯一确定因特网中的TCP连接双方。源端口和目的端口的长度都为2个字节。
 - 序号字段：标识TCP报文段数据部分第一个字节在源端口发送字节流中的位置，长度为4个字节。
 - 确认号字段：标识目的端口希望接收到的下一个TCP报文段数据部分第一个字节的序号，其长度为4个字节。
- 

- 首部长度字段：标识TCP报文段首部的长度，长度为4个比特位。首部长度字段的度量单位是4字节，TCP报文段首部的最大长度为60字节。
 - 保留字段：为今后应用保留，长度为6个比特位，通常设置为0。
 - 标志位字段：包含有6个标志位，从左到右分别为URG、ACK、PSH、RST、SYN和FIN，它们用来标识不同类型的TCP报文段。长度为6个比特位。
- 

标志位字段的含义及使用说明

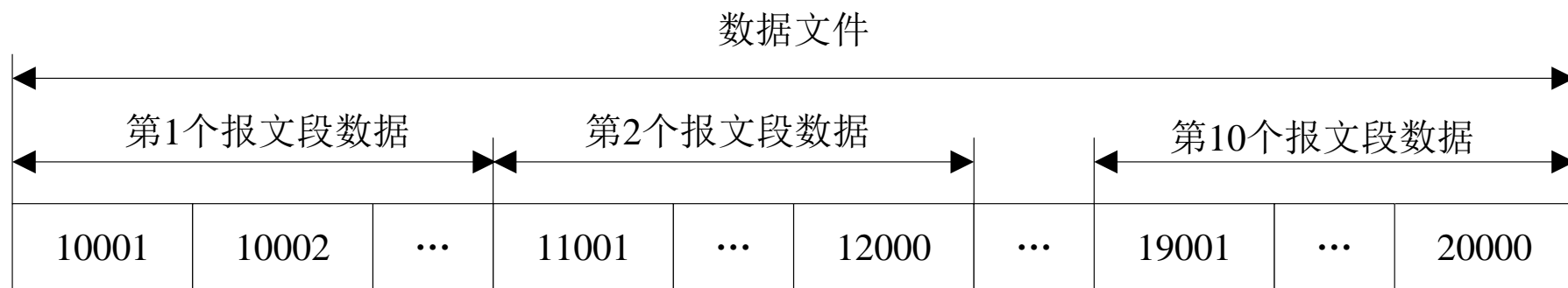
名称	含义	使用说明
URG	设URG=1时，表明报文段中有紧急数据，这时紧急指针字段有效。	紧急数据插在报文段数据的前面， 可以先被处理。
ACK	设ACK=1时，表明确认号字段有效	用于设置确认号字段
PSH	设PSH=1时，表明请求推操作有效	发送方设置PSH=1，是让目的端口尽快将接收到的TCP报文段交付（推送）给应用层，而不必在缓冲中排队
RST	设RST=1时，表明连接复位操作有效	用于设置连接复位，当TCP连接出现错误时，需要断开连接，再重新建立连接
SYN	设SYN=1时， ACK=0时，表示请求建立TCP连接； 当SYN=1， ACK=1时，表示响应建立TCP连接	当TCP建立连接时，用于设置同步序号，通常与ACK标志位配合使用
FIN	设FIN=1时，表明源端口的字节流发送完毕，并请求断开连接	用于连接释放

- 窗口字段： 2个字节， 进行流量控制。接收方通知发送方自己目前能够接收数据量（由缓冲空间限制）， 发送方据此设置发送窗口。
- 校验和字段： TCP报文段的差错检测， 长度为2个字节。TCP校验和计算方法与UDP校验和计算方法相同。
- 紧急指针字段与标志位字段中的URG标志位配合使用， 长度为2个字节。
- 选项： 为TCP提供扩展功能
- 填充字段： 当选项部分长度不是4个字节的整数倍时， 需要进行填充。

6.3.3 TCP序号与确认

- 在TCP传输的字节流中，TCP为每个字节分配一个序号，序号为32位无符号数，在区间为 $0 \sim 2^{32}-1$ 中循环使用。
- 在TCP连接建立时，源端口与目的端口需要商定一个初始序号，以后发送的报文段中序号的设置是初始序号的延续。


TCP报文段序号分配示例




- 在TCP报文段传输过程中，由于网络拥塞，可能会出现报文段丢失的情况，这就需要目的端口对所接收到的正确的报文段进行确认，以便通知源端口所发送的报文段的目前状态。
- TCP支持选择确认、累积确认和捎带确认。



6.3.4 TCP重传机制

- 若TCP报文段在传输过程中丢失或产生差错，将采用重传机制重传此类报文段。
 - **缓存-定时-超时重传：**源端口在发送一个TCP报文段时，首先将复制一个副本存放在缓冲区内，并启动计时器；如果在规定的时间内没有收到来自目的端口的确认信息，则判定该报文段丢失或产生差错，并重传该报文段副本。
- 

- 计时器超时时间（TimeOut）的设置：取决于TCP报文段传输的往返时间（Round Trip Time, RTT），即从源端口到达目的端口以及源端口收到目的端口的确认信息的所经历的总时间。
 - 由于因特网环境的不确定性，RTT是变化的。
 - TCP采用了自适应重传算法以适应网络的不确定性。
- 

- 自适应重传算法的基本思想：TCP监视每一个连接的性能，根据相应连接RTT的变化随时调整TimeOut的设置，以适应因特网环境的变化。

1. 计算平滑的RTT

$$RTT = \alpha * Old_RTT + (1 - \alpha) * New_SampleRTT$$

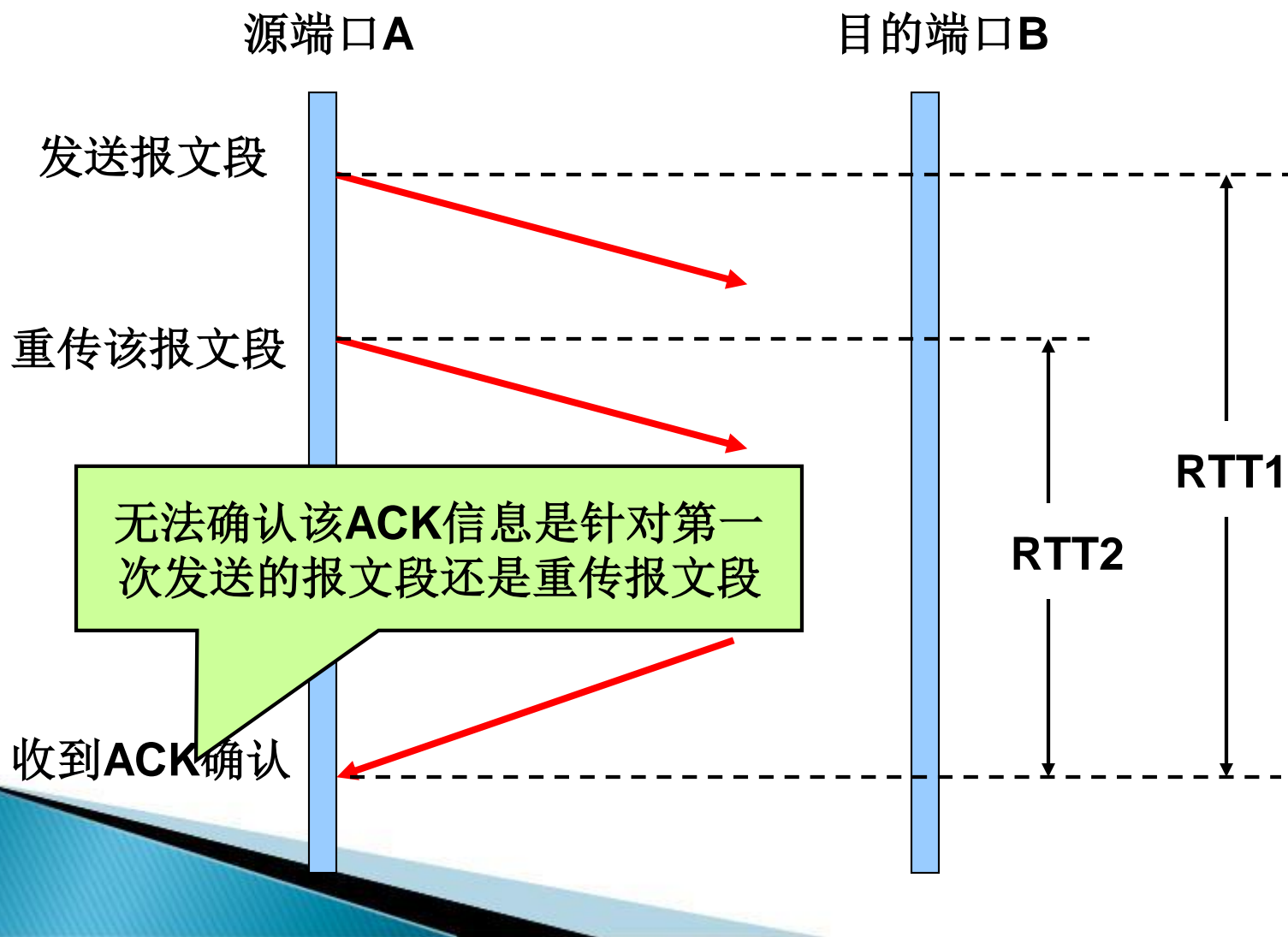
$$(0 \leq \alpha < 1)$$

2. 超时时间略大于RTT

$$TimeOut = \beta * RTT \quad (\beta > 1)$$

$$(\beta \text{建议值为} 2)$$

TCP确认的二义性



- 问题：发生重传时新的RTT不可知。
- Karn算法：
不采用发生重传时测得的RTT来重新计算往返时间估计值。
- 新问题：网络延时增大没有及时反映在超时值上，会导致更多的重传发生。
- 对Karn算法的修正：发生重传时，增加超时值：
$$\text{New_TimeOut} = \gamma * \text{TimeOut} \quad (\gamma > 1)$$


(γ 取值通常为2)

6.3.5 TCP连接管理

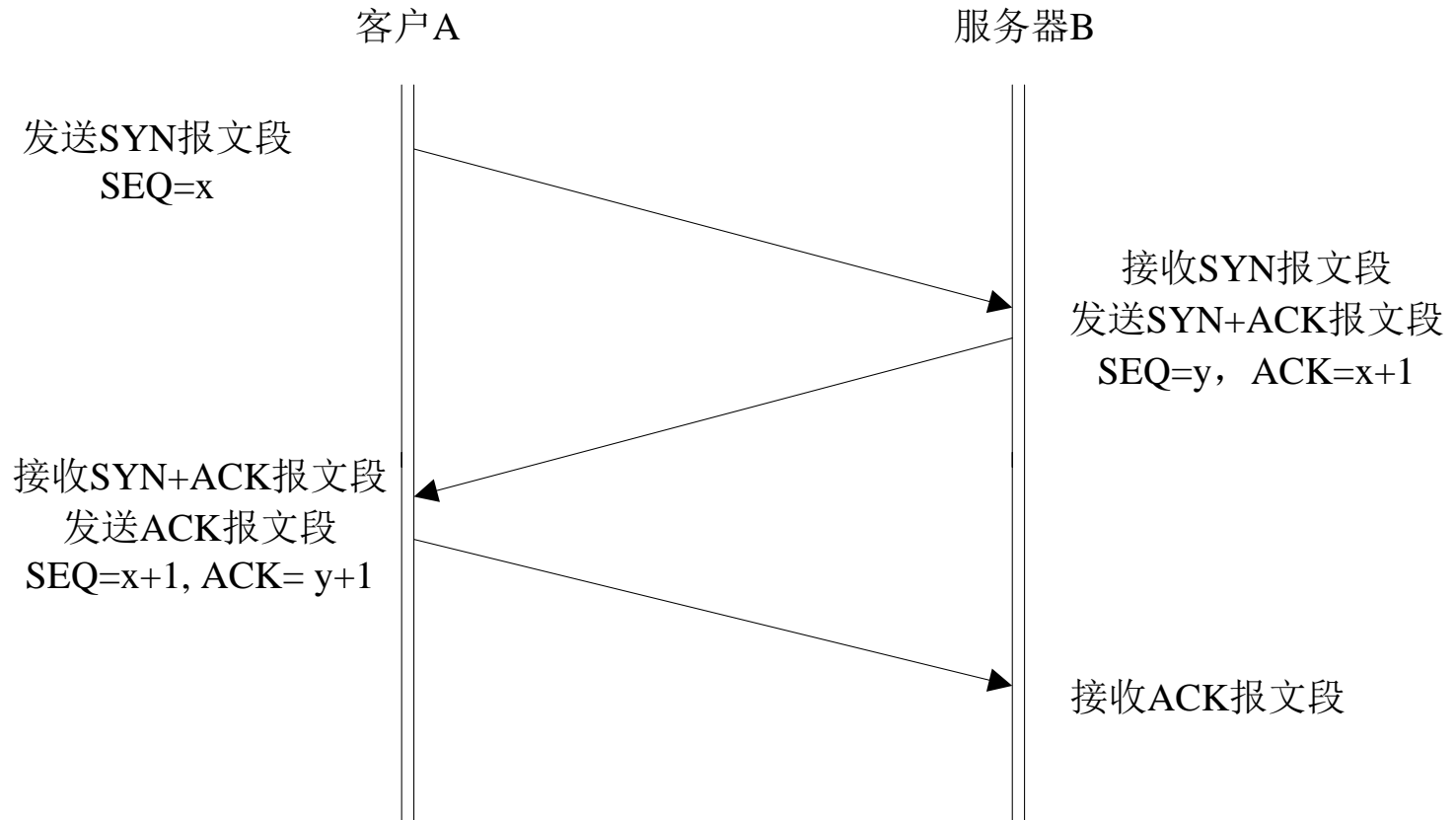
- TCP是一个面向连接的协议，通信双方不论哪一方发送报文段，都必须首先建立一条连接，并在双方数据通信结束后关闭连接。



1. 建立连接

- TCP使用三次握手建立连接。
 - 在建立连接过程中，请求发起连接建立的应用进程被称为客户，等待建立连接的应用进程被称为服务器。
 - 为了建立连接，服务器执行被动打开命令等待连接请求的到达；客户则执行主动打开命令，并指明它想要连接到的服务器的IP地址和端口号，等待服务器的响应。
- 

三次握手建立TCP连接



2. 关闭连接

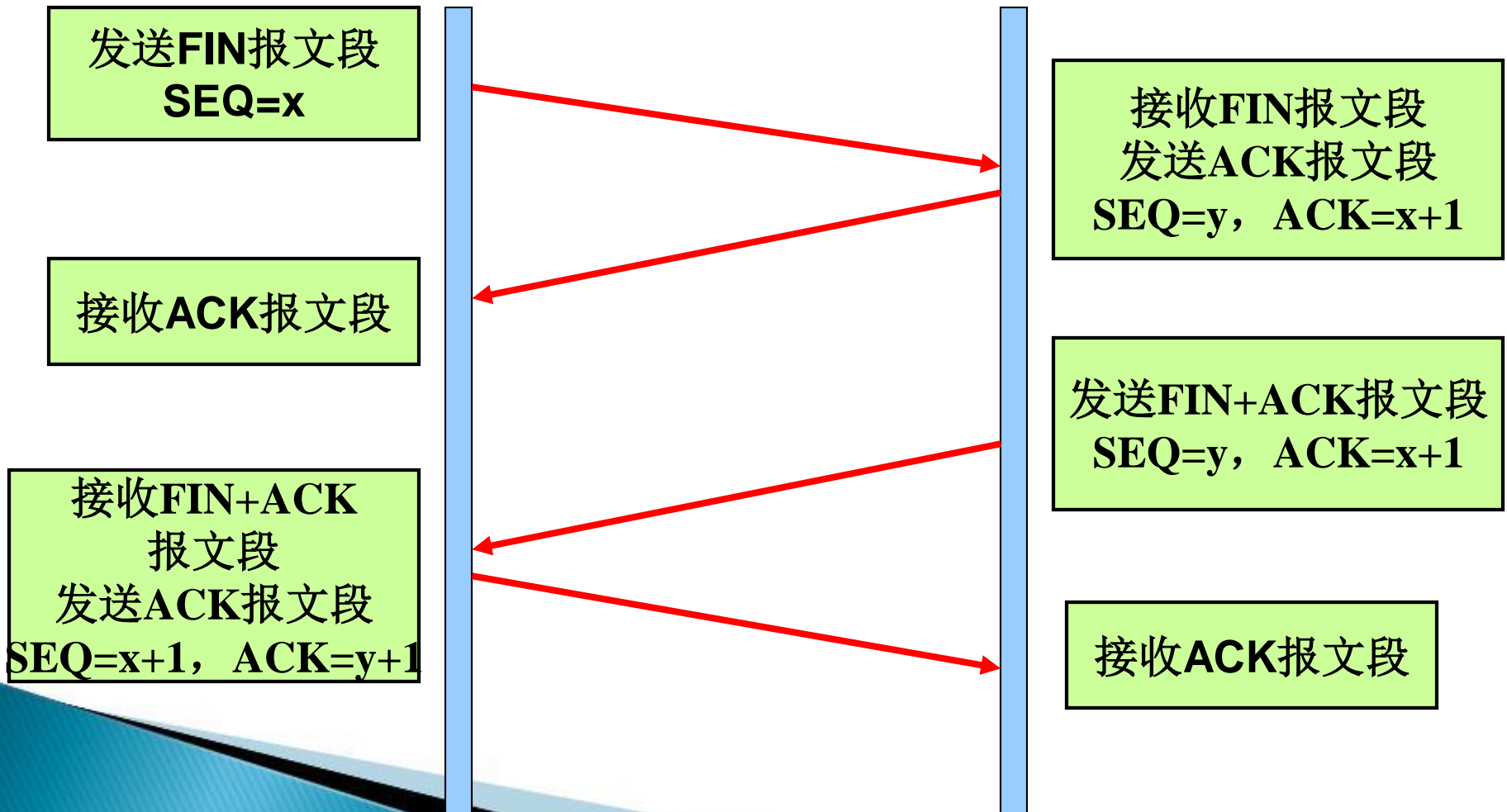
- TCP通信是全双工通信，通信双方中的任何一方在数据传输结束后，都可以向对方发起关闭连接的请求，以结束一个方向的连接。



关闭一个TCP连接

客户A

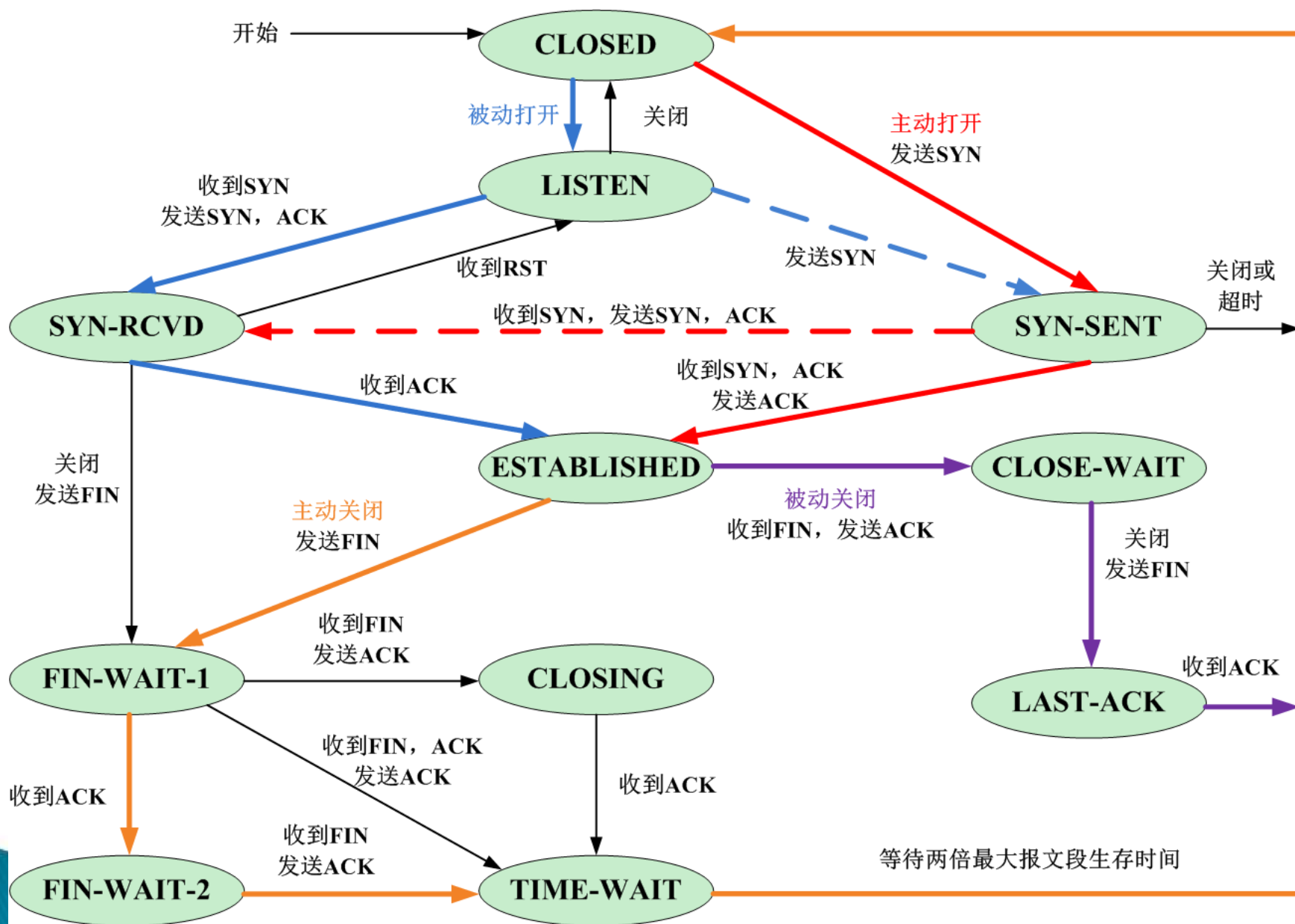
服务器B



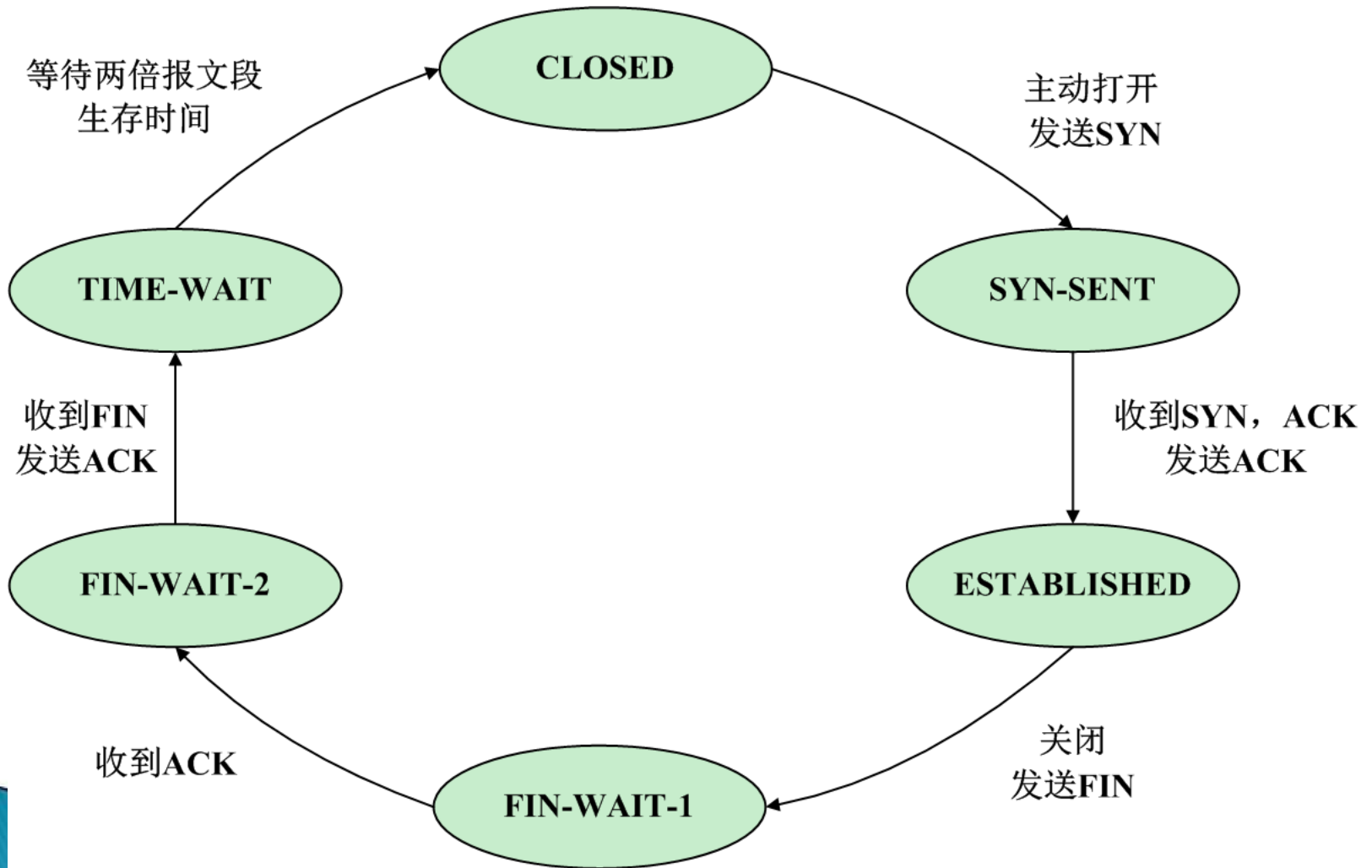
3. 连接状态管理模型

- TCP采用有限状态机的连接状态管理模型，该模型能够解释TCP连接中各种可能的状态以及状态下一步可能发生的转换。
- TCP有限状态机中共包含有11种状态，每个TCP连接都从CLOSED状态开始。

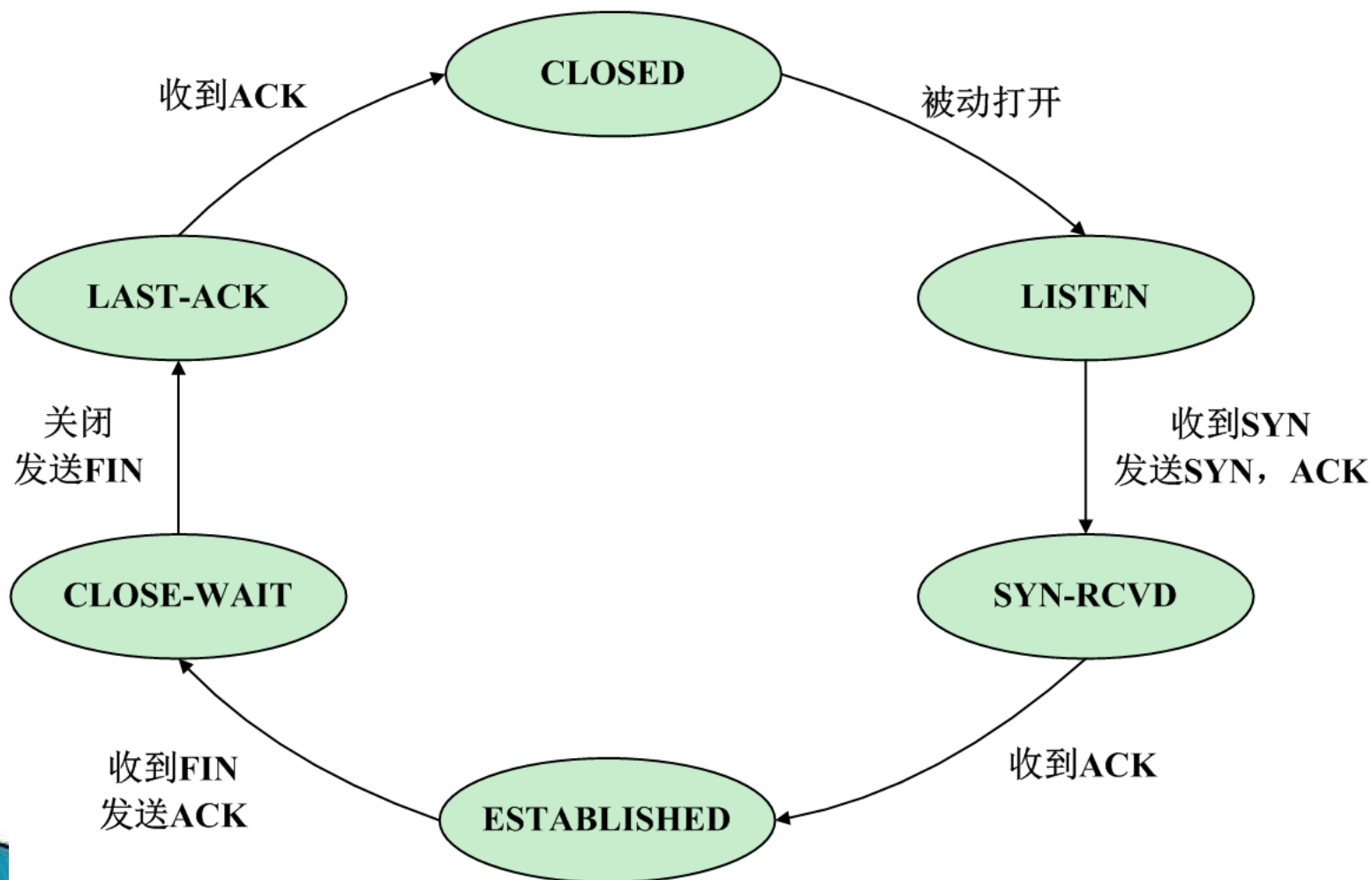





发起连接请求方建立和关闭一个TCP连接状态转换过程



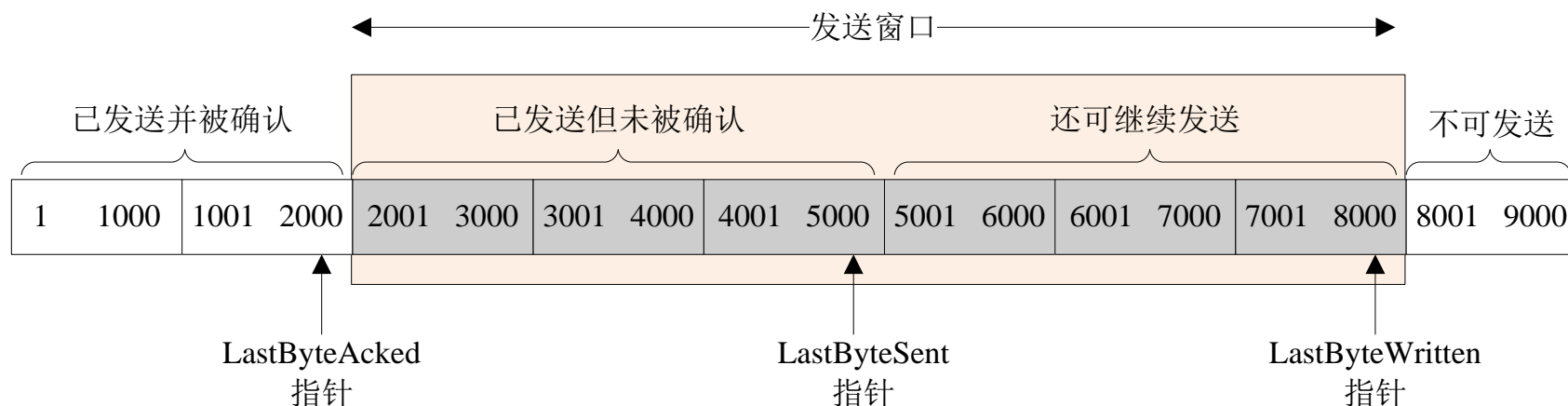
被请求方建立和关闭一个TCP连接状态转换过程



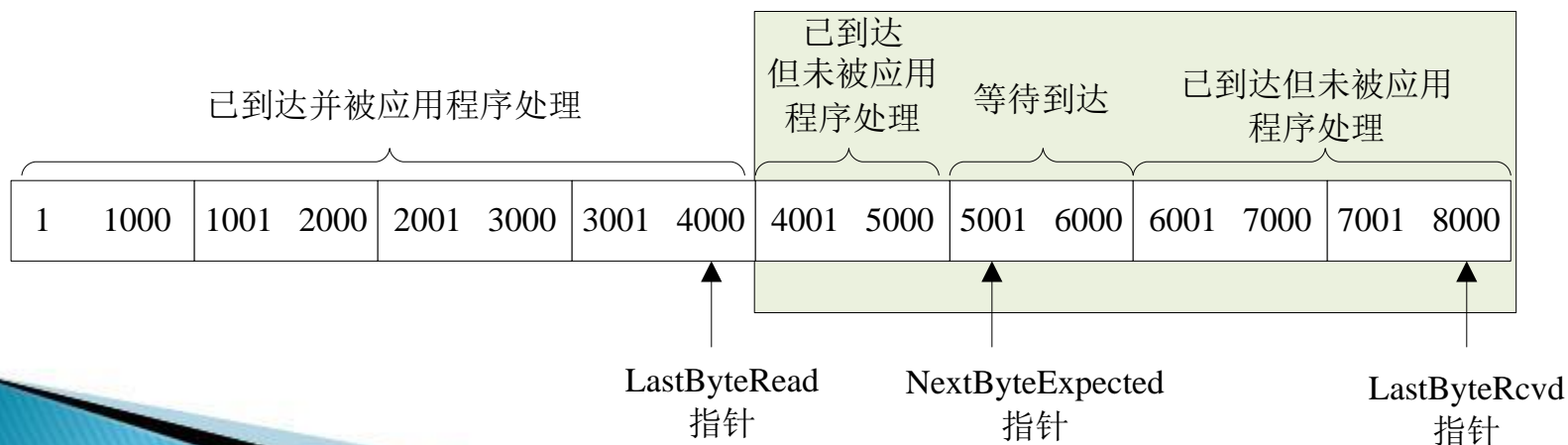
6.3.6 TCP流量控制


- 由于TCP通信双方缓冲空间分配以及数据处理速度的不同，可能会造成接收方数据溢出的情况，这就需要一种控制机制能够协调通信双方的数据流量。
 - TCP 使用滑动窗口机制解决上述问题。
 - 每个TCP连接维持两个窗口，即发送窗口和接收窗口，窗口大小的单位是字节。
 - 接收的报文段可能乱序到达。
- 

TCP发送窗口



TCP接收窗口



- 在TCP流量控制过程中，发送窗口的大小随着接收方发布的窗口通告值进行调整。
 - 窗口通告值增大时，发送方扩大发送窗口的大小，以便发送更多的数据。
 - 窗口通告值减小时，发送方缩小发送窗口的大小，以便接收方能够来得及接收数据。
 - 窗口通告值减小至零时，发送方将停止发送数据，直到窗口通告值重新调整为大于零的数值。
- 


TCP流量控制

发送方A

接收方B

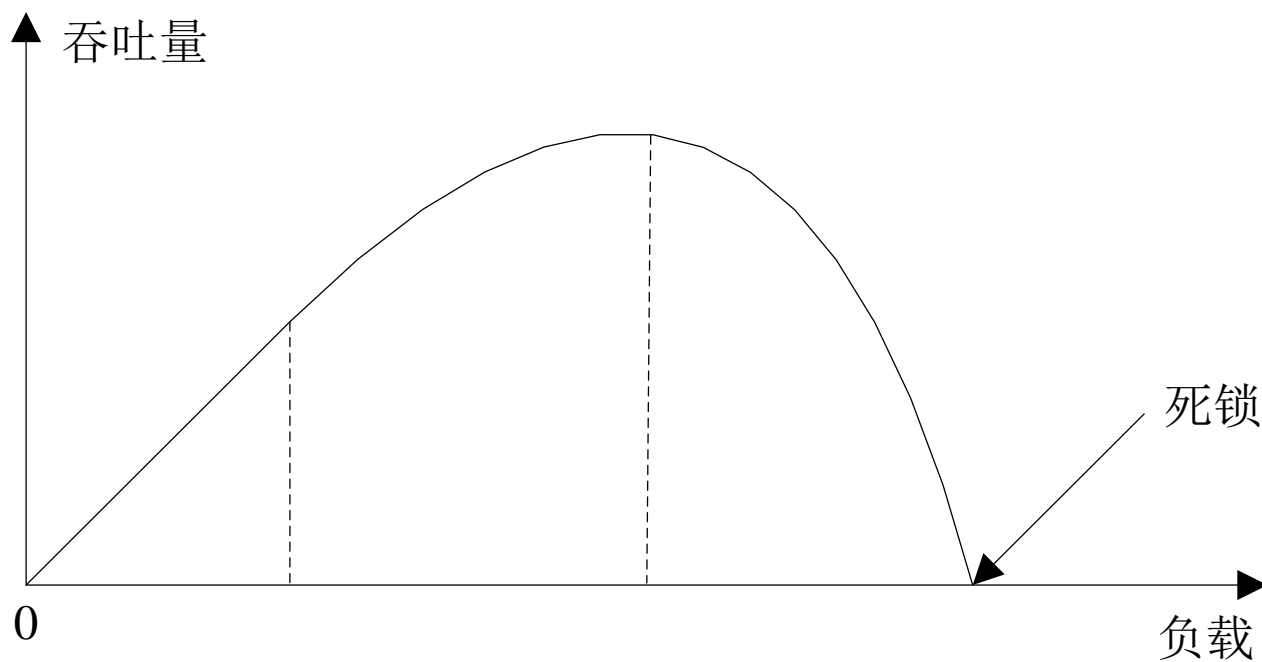


6.3.7 TCP拥塞控制

- 拥塞（congestion）是指因特网中的数据报过多，超过了中间结点的最大容量，导致网络性能急速下降的现象。
 - 拥塞控制（congestion control）算法主要用于避免拥塞现象发生。
 - TCP通常综合采用慢开始、拥塞避免、快速重传和快速恢复等拥塞控制算法。
 - TCP流量控制只考虑接收方的接收能力。
- 

拥塞发生过程

无拥塞时：吞吐量与网络负载呈线性增长的关系；
轻度拥塞：吞吐量随着网络负载的增长而缓慢增长；
严重拥塞：吞吐量随着网络负载的增长而急剧降低直至死锁。



拥塞窗口

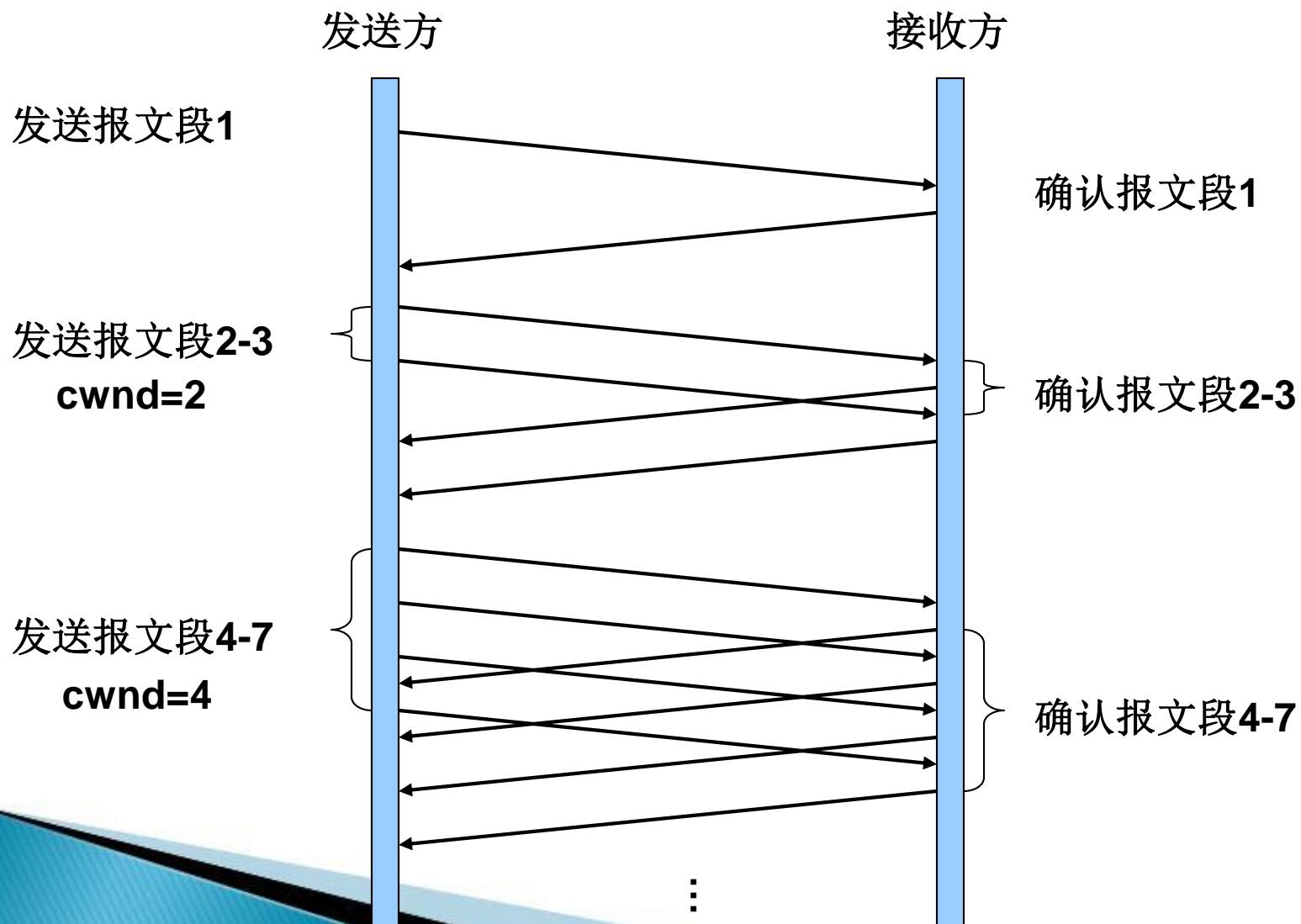
- 接收窗口rwnd（receiver window）：流量控制中接收方的通告值，反映接收方的接受能力，不能体现中间结点的处理能力。
- 拥塞窗口 cwnd（Congestion window）：由发送方根据网络的情况设置，表示发送方允许发送的最大报文段。



1. 慢开始和拥塞避免

- 慢开始算法要点：
 - ✓ 建立连接后，拥塞窗口的初始值设置为1（1个报文段）。
 - ✓ 发送方取rwnd和cwnd中的较小值作为当前发送窗口值进行数据报发送。
 - ✓ 每收到一个新的报文段的确认，将拥塞窗口的大小增加一个报文段，cwnd以指数方式快速地增长。
 - 每收到1个确认，cwnd值就增加1（cwnd从1到2）
 - 收到2个确认，cwnd值就增加2（cwnd从2到4）

慢开始算法



慢开始阈值

- 慢开始算法中的拥塞窗口cwnd会以指数方式快速增长。
- 为避免cwnd过快增长引起网络拥塞， 设置**慢开始阈值**（ssthresh）。
 - ✓ $cwnd < ssthresh$ 时采用慢开始算法；
 - ✓ $cwnd \geq ssthresh$ 时采用拥塞避免算法， 减慢窗口增长速度。

拥塞避免算法

当 $\text{cwnd} \geq \text{ssthresh}$ 后，启用拥塞避免算法：

- 每经过一个往返时延RTT，只有当发送方收到对所有报文段的确认后，才将拥塞窗口的大小增加一个报文段。
- 加法增加（Additive Increase）策略， cwnd 按照线性方式缓慢增长，与慢开始算法相比，其增长速度放慢，直到网络出现拥塞。

设 $ssthresh=4$

发送方

接收方

发送报文段1

确认报文段1

发送报文段2-3

确认报文段2-3

$cwnd=ssthresh$

发送报文段4-7

确认报文段4-7

$cwnd=5$

发送报文段8-12

确认报文段8-12

$cwnd=6$

发送报文段13-18

确认报文段13-18

慢开始算法

拥塞避免算法

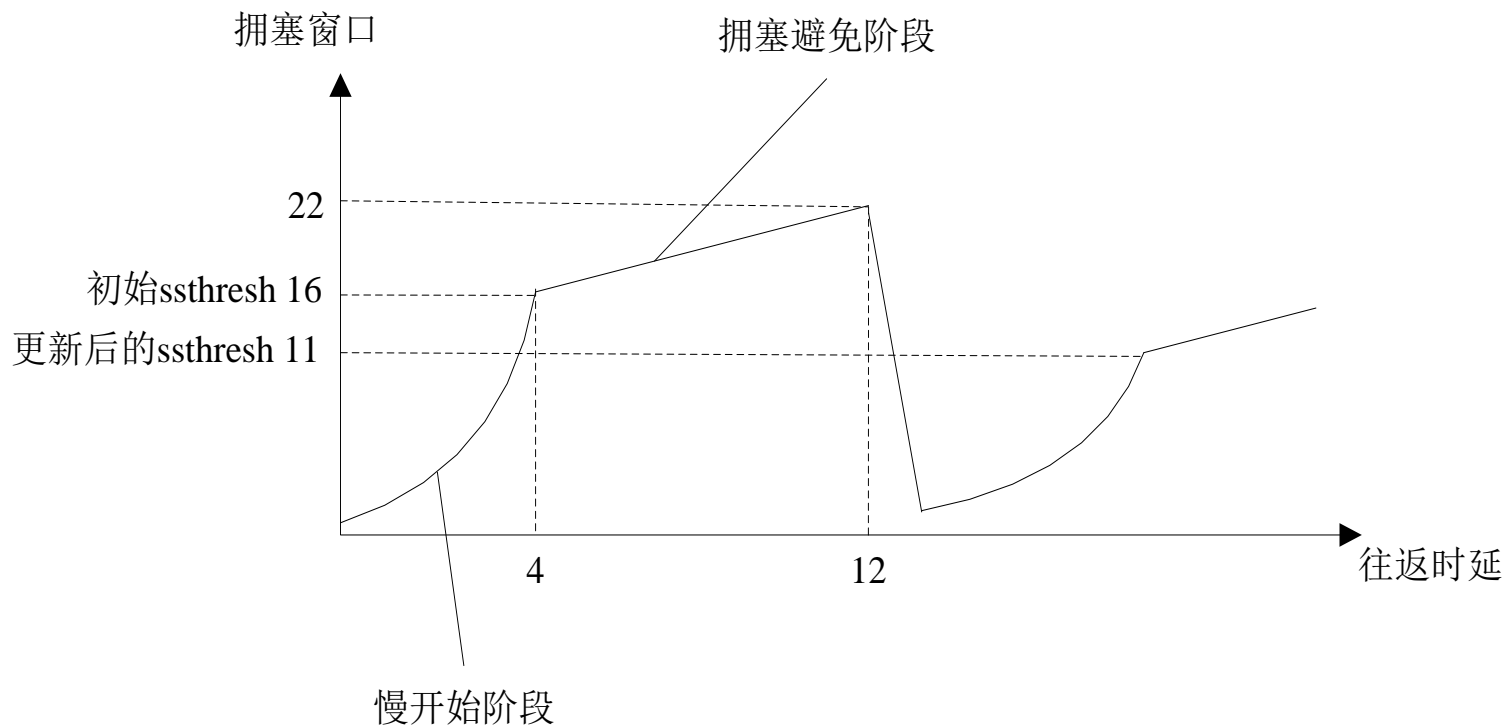


乘法减小策略

若有报文段丢失，则判定网络发生拥塞，此时不论是处于慢开始阶段还是拥塞避免阶段，发送方将采用乘法减小（Multiplicative Decrease）的策略：

- 慢开始阈值 $ssthresh$ 减为当前 $cwnd$ 的一半（但不能小于2）。
- $cwnd$ 重新设为1，启动慢开始算法。
- $cwnd$ 增长到新的慢开始阈值 $ssthresh$ 大小时，进入拥塞避免阶段。

慢开始和拥塞避免算法示例

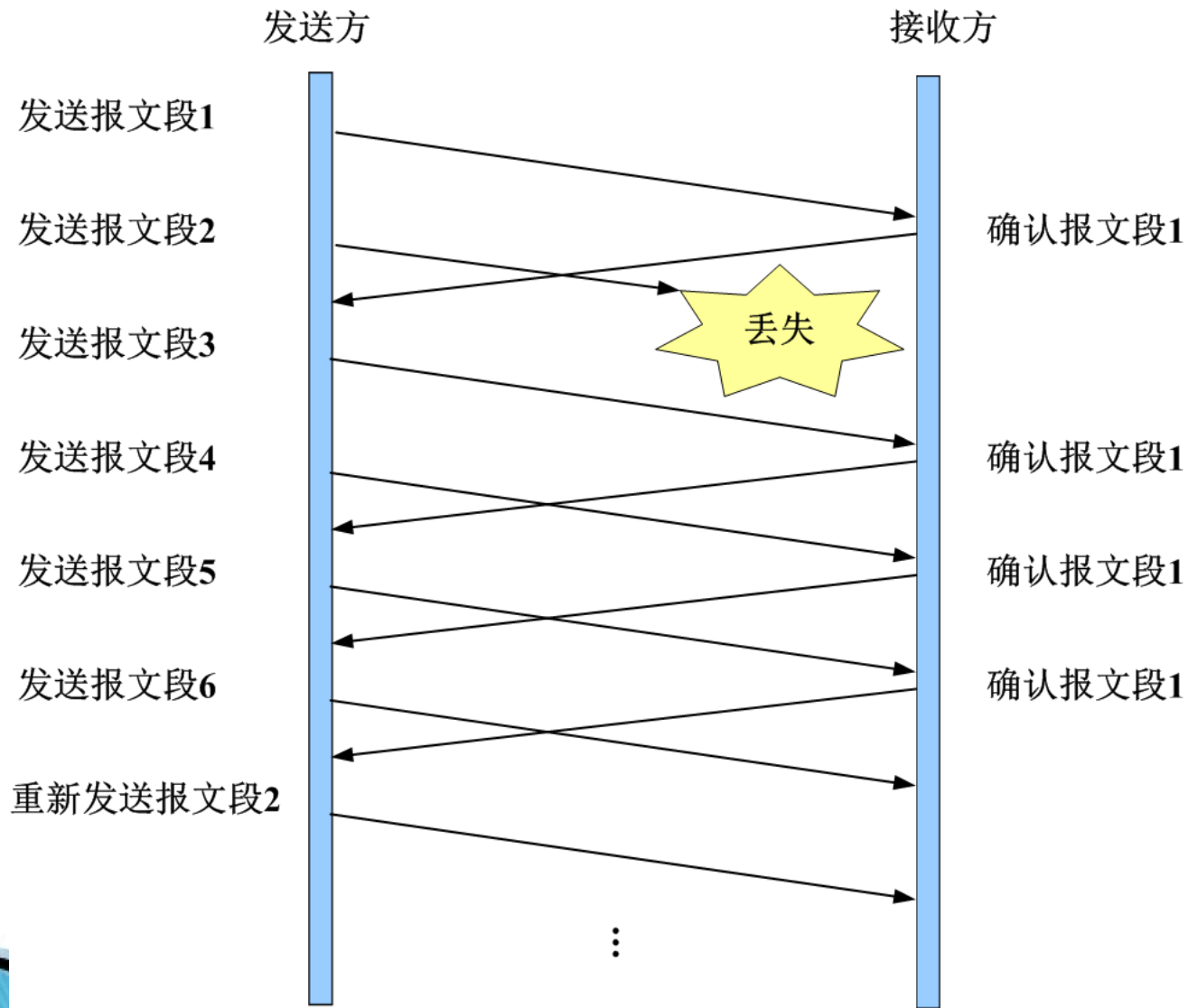


2. 快速重传算法

- 接收方收到乱序到达的TCP报文段后，立即发送确认报文段，通告期待接收的报文段最小序号。
- 若发送方连续收到三个重复的确认报文段，认定该报文段丢失。无论计时器时间是否结束，立即重传该报文段。




快速重传算法示意图



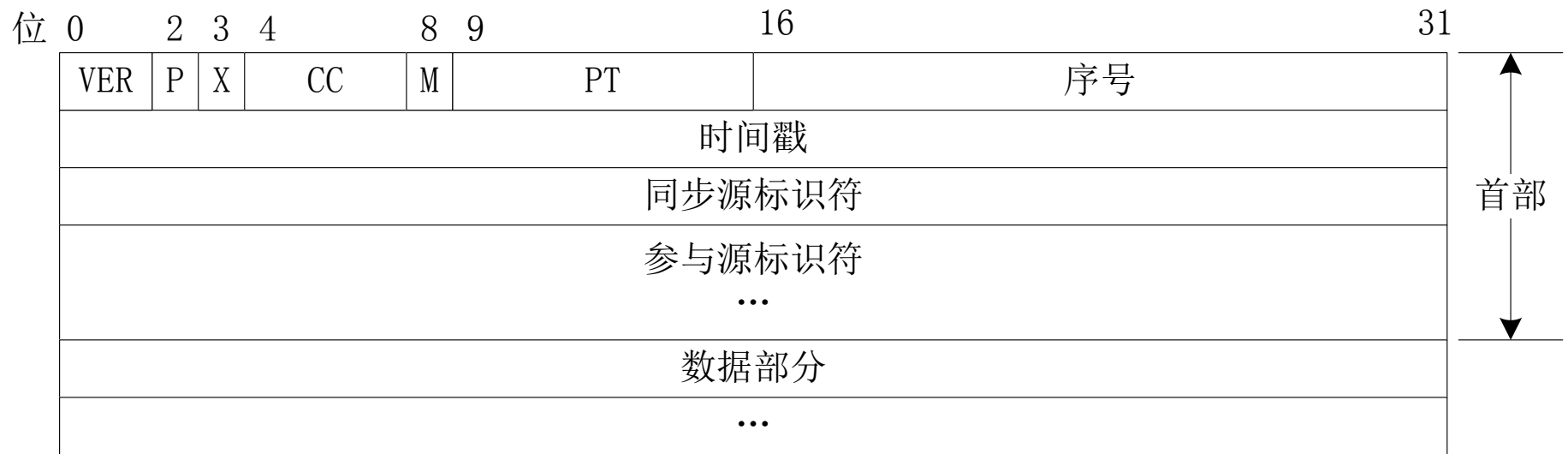
快速恢复算法


- 发送方连续收到三个重复的确认报文段，启动拥塞避免算法：
 - 将慢开始阈值 $ssthresh$ 减半
 - 将拥塞窗口 $cwnd$ 的大小调整为更新后的 $ssthresh$ 的值
 - 启动拥塞避免算法，之后拥塞窗口 $cwnd$ 的值按照线性方式增长
- 不启动慢开始算法的原因：只有一个报文段丢失，其它报文段正确接收，说明当前网络并没有发生严重的拥塞。

6.4 用于多媒体传输的实时传输/ 传输控制协议

- RTP: 用于音视频等多媒体数据流的传输协议。
 - RTCP: 在RTP会话期间, 为RTP提供带外控制。
 - RTP/ RTCP应用在UDP协议之上。
 - IETF RFC 3550对RTP / RTCP做出了明确的定义和规范。
- 

RTP报文格式



- VER: 2位, 协议的版本号, 当前版本号为2。
 - P: 1位, 填充位, 如果有效载荷后面需要填充, 则该位置位。
 - X: 1位, 扩展位, 如果该位设置, 则在固定首部后跟随有一个扩展首部。
 - CC: 4位, 参与源计数, 指出在固定首部后的参与源的数目。
 - M: 1位, 标记位, 含义取决于具体的应用。
 - PT: 7位, 定义有效载荷类型。
- 

- 序号：16位，RTP报文序号，接收端可通过序号来检查报文段的丢失情况，并可针对乱序的报文段进行重新按序排列。
- 时间戳：长度为32位，给出RTP报文段中第一个字节的采样时间。可用于接收端对时延抖动的消除和对具体应用的同步。




- 同步源标识符（Synchronization SouRCe identifier, SSRC）：32位，标识数据流的源，随每个数据流的开始时随机产生，每一个数据流都有一个唯一的同步源标识符。
- 参与源标识符（Contributing SouRCe identifier, CSRC）：参与源个数不定，但最多不超过15个，每个为32位。该字段由混合器插入，标识混合器中不同的数据流，混合器将多个数据流合并起来，形成一个数据流进行传输。



封装到UDP中的RTP报文段




RTCP

- RTCP应用在UDP协议之上，其主要功能是提供RTP服务质量的反馈，为RTP提供带外控制。
 - 在RTP会话期间，同一个RTP会话的参与者采用IP多播的方式定期发送RTCP报文。
 - IETF RFC 3550对RTCP做出了明确的定义和规范。
- 

RTCP五种基本报文类型

类型	意义
200	发送端报告（ SR ）
201	接收端报告（ RR ）
202	源描述（ SDES ）
203	结束（ BYE ）
204	特定应用（ APP ）

- **SR**: 针对所发送的每个**RTP**数据流提供信息，主要包含：
 - **RTP**流的**SSRC**
 - 流中最新分组的**RTP**时间戳
 - 绝对时钟（**NTP**）时间戳，用来同步一个**RTP**会话中的不同媒体流（如视频流和音频流）。也可以根据发送和接收双方的**NTP**时间戳计算往返时延。
 - 流所发送的分组数和字节数等。
- 

- **RR:** 接收端定期向发送端报告，针对每个**RTP**媒体数据流提供报告信息，以便发送端了解接收状况，并根据实际情况调整发送策略。
- 主要包含：
 - **RTP**流的**SSRC**
 - 流的分组丢失数、丢失率、时延抖动
 - 流中最新收到的**RTP**分组的序号等。

- 源描述（SDES）：用于描述RTP流的发送端（源），基本信息包括：
 - 用于标识发送方的规范名称（可按照某种格式自动生成，如[doe@192.0.2.89](#)等）
 - 创建该RTP流的应用程序或工具的名称
 - 用户名、用户的电子邮件地址、电话号码等信息。
- 结束报文（BYE）报文：表示会话成员退出，该数据源关闭。
- 特定应用（APP）报文：由应用程序自己定义。

课后思考题

1. 在因特网中，用什么来**唯一地**标识特定的网络应用程序进程？
2. 从目标和手段两方面分析：流量控制和拥塞控制有什么不同？

