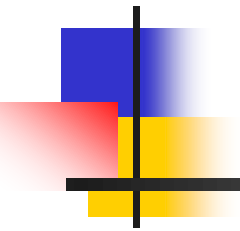


TCP与可靠传输

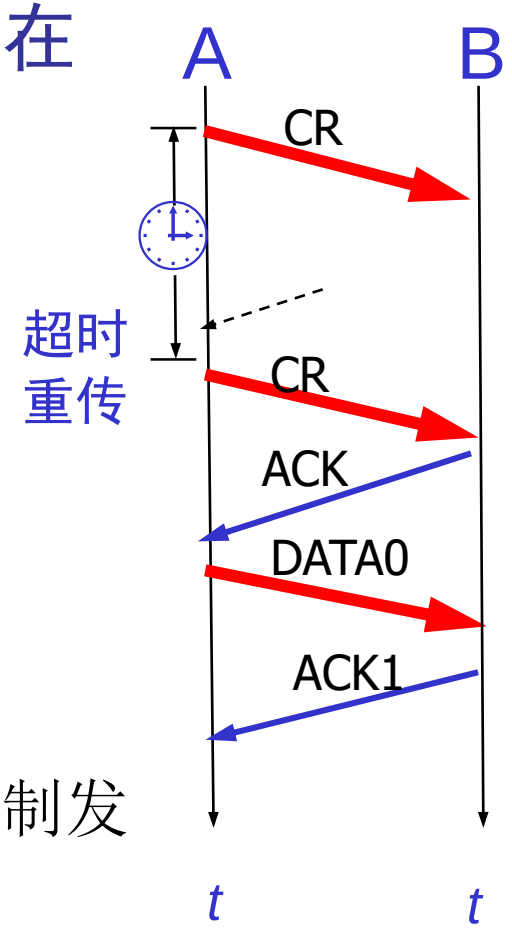
刘志敏
liuzm@pku.edu.cn

提纲

- 可靠传输机制
  - 超时重传
  - 肯定确认
- 面向连接的连接管理
  - 二次握手
  - 三次握手
- TCP协议
  - 三次握手
  - 滑动窗口与流量控制

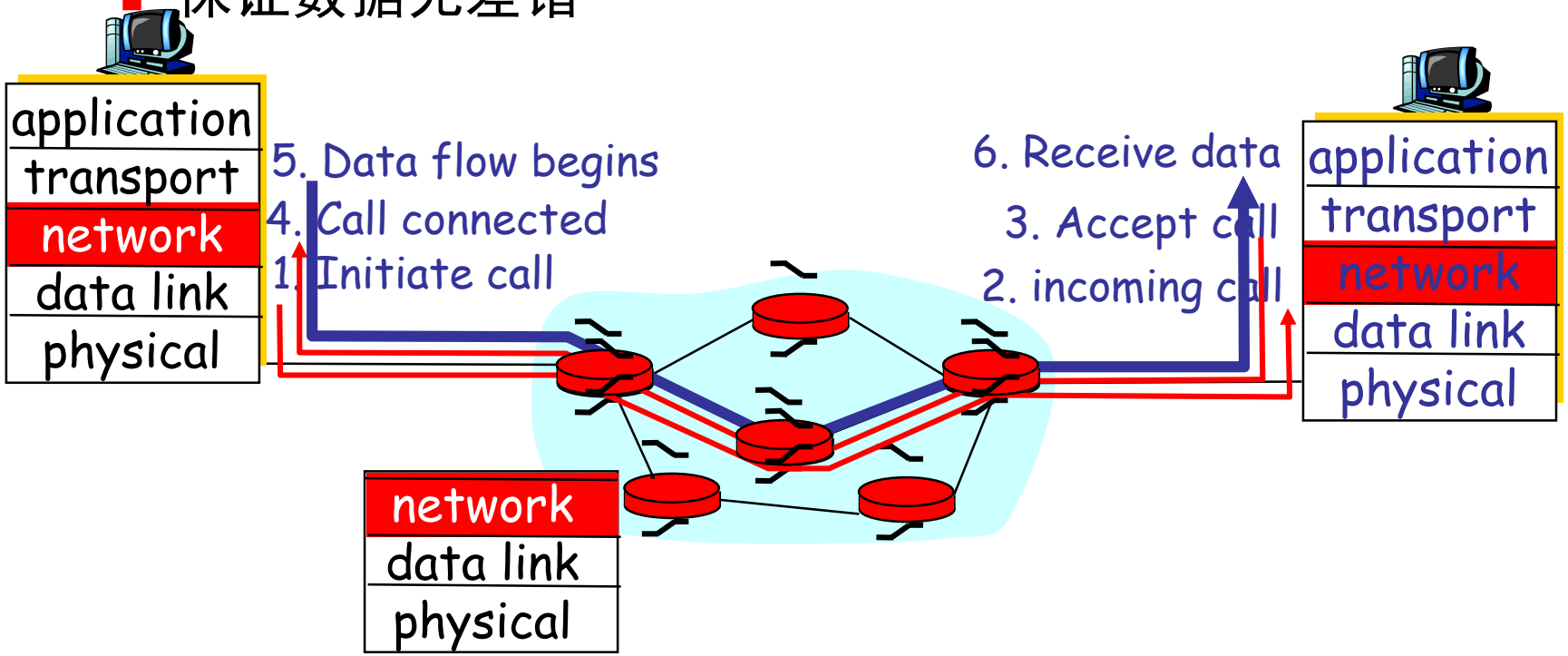
回顾：链路层的连接管理与可靠传输

- 链路层：相邻结点间的链路，存在错帧、丢帧，但无错序问题
- 建立连接
  - 连接请求CR，收到确认ACK
  - 二次握手，保证结点间的状态同步
- 差错控制与流量控制
  - 帧校验FCS、帧序号
  - 定时及超时重传
  - 滑动窗口：用接收方的确认帧，控制发送方的发送窗口



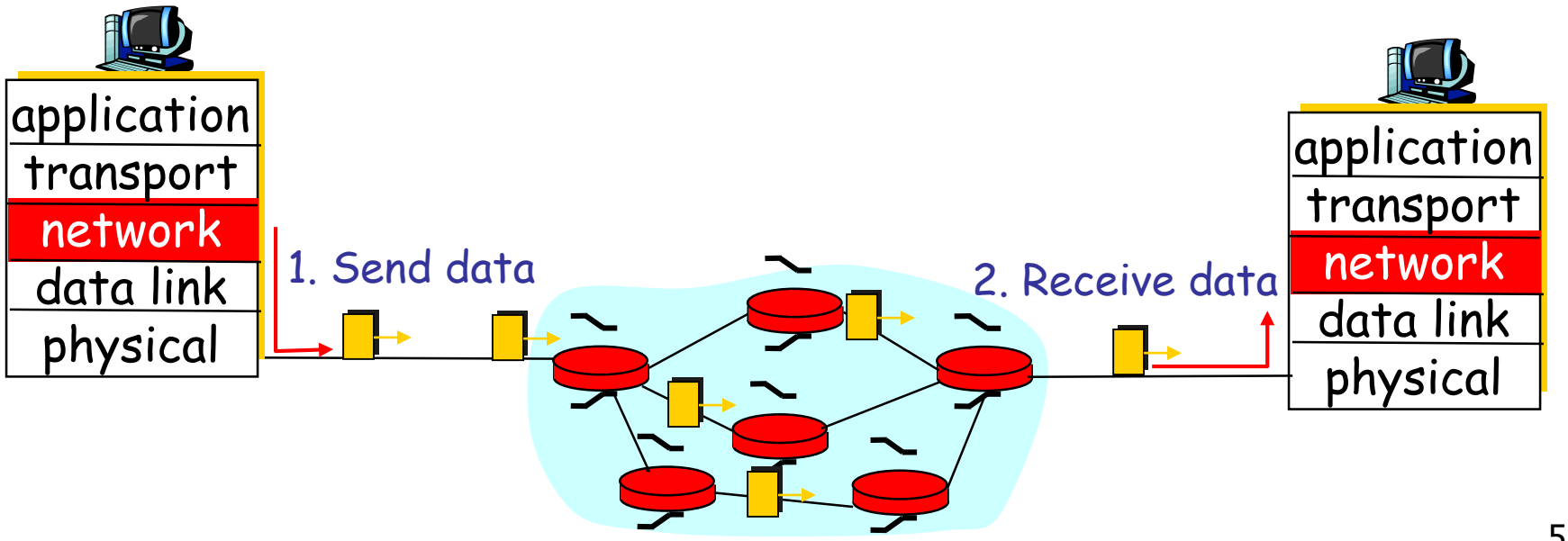
回顾：网络层的连接管理与可靠传输

- 网络层：非相邻结点之间的分组交换
- 若网络层提供面向连接的服务
  - 二次握手，建立连接
  - 数据沿着同一路径传输，流量控制与差错控制
  - 保证数据无差错



回顾：互联网网络层的数据报服务

- 网络层提供无连接的服务，没有呼叫建立过程
- 基于目的地址对每个分组进行路由选择
  - 同一源地址——目的地址的分组，可以选择不同路径
- 保证无差错，但存在网络拥塞、丢失、错序等问题
- 传输层上提供面向连接的服务，可以采用二次握手吗？



连接建立

发送连接请求CR，等待确认ACK

网络拥塞，则数据被迫多次重传

因网络拥塞导致的分组丢失、延迟等，例如
　此前很早发出的CR又出现在接收端了

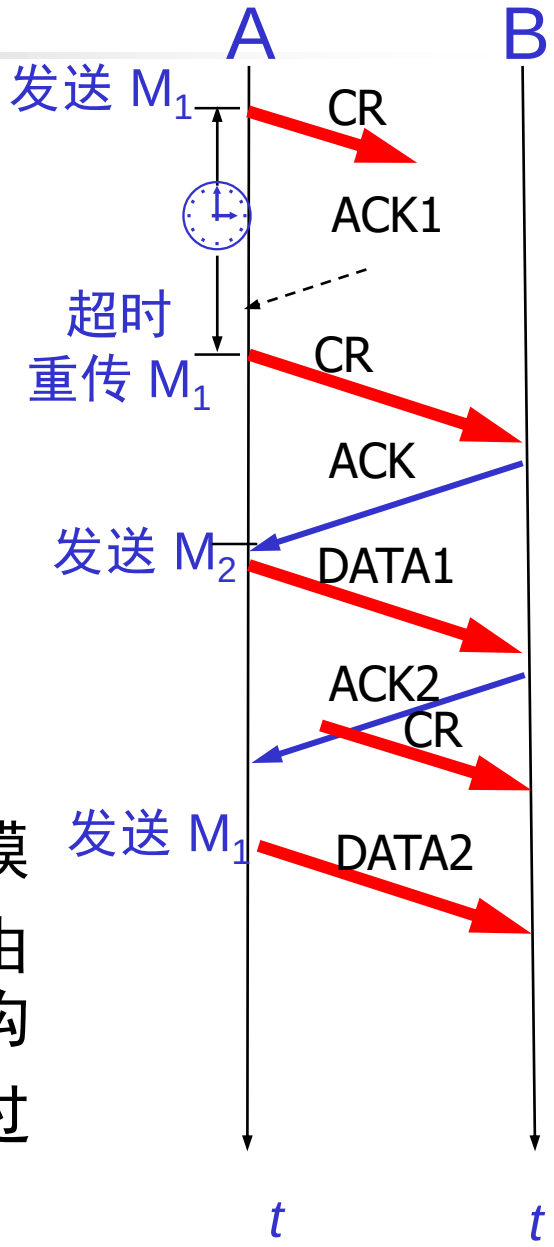
接收重复数据导致严重问题，如银行付款请求

如何避免重复，拒绝重复数据？

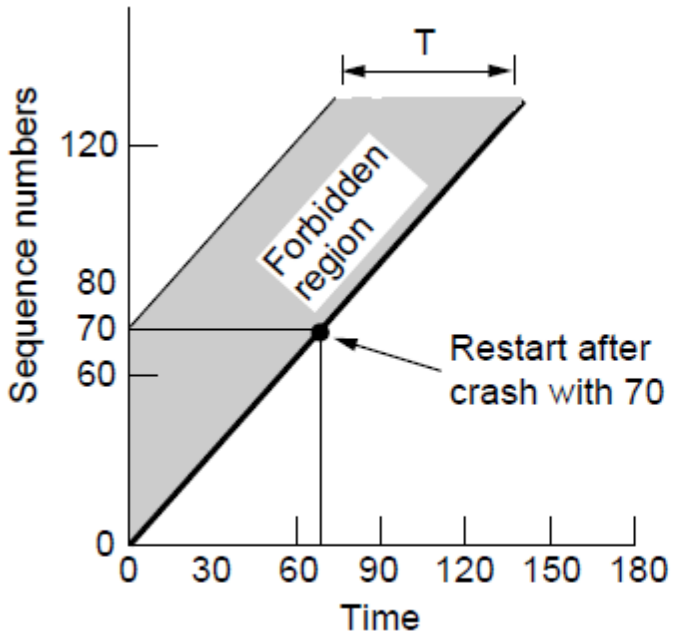
需要严格限制分组的生存时间！

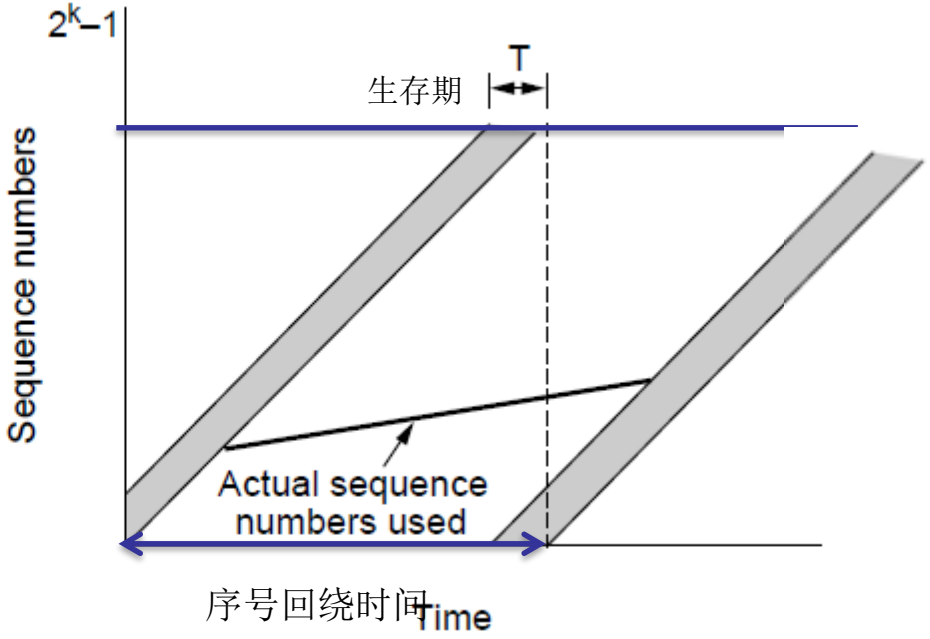
可选方案：

- 规定数据报最大延迟→限制了网络规模
- 在数据段中设置跳数计数器，需要路由器检测传输层数据段→破坏了分层结构
- 每个报文段一个时间戳，路由器丢弃过时的报文段→ 要求路由器时钟同步



连接建立(2)





设段的最大生存期为T（例如120s），源端用序号作段标记，在T内只能有一个某序号的段。收端根据序号接收新的并丢弃重复的段。

如何保证机器崩溃内存数据丢失情况下在T内序号不重用？方法是基于系统时钟（因机器崩溃但系统时钟继续）设置初始序号。

序号空间应足够大，确保序号回绕时旧序号的段已经消失，设S为最大序号（2^K），C为发送数率，序号回绕时间为T_0，则$T_0 \times C=S$

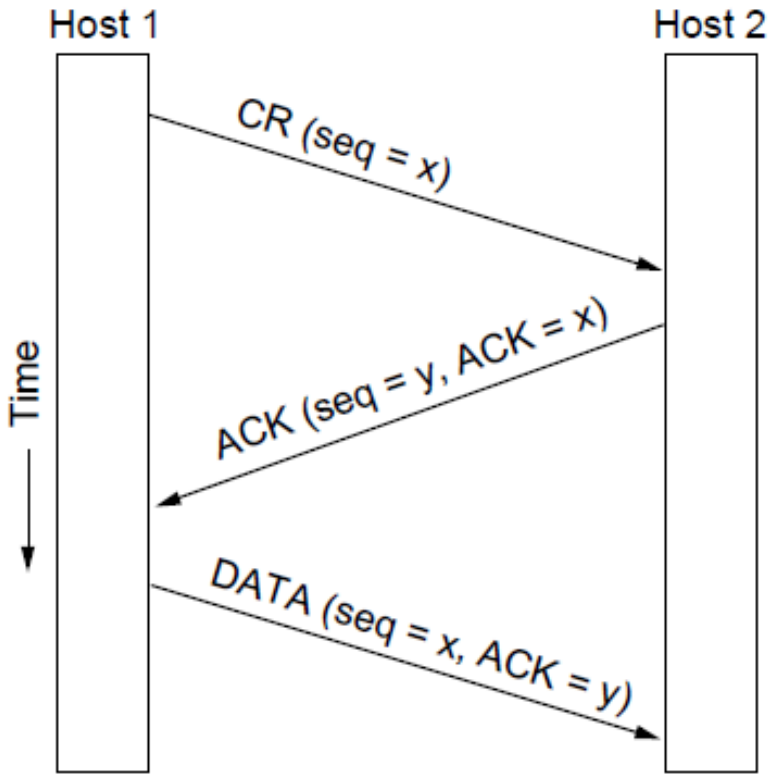
例题

- 为了避免序号回绕，采用64位序号，光纤链路的数据速率为75Tbps。问未来的75Tbps网络采用64位序号，问序号回绕的时间是多少？假设每个字节都有一个序号。
- 最大序号$S=2^{64}=10^{19.2}$，与表示序号的位数有关
- 发送速率C= 75Tbps=$75/8\times10^{12}$Bps
- 序号回绕时间$T_0<S/C=37\times8$小时
- 若报文段的序号采用32位，则$T_0=10^{9.6}/（75/8\times10^{12}）=?$

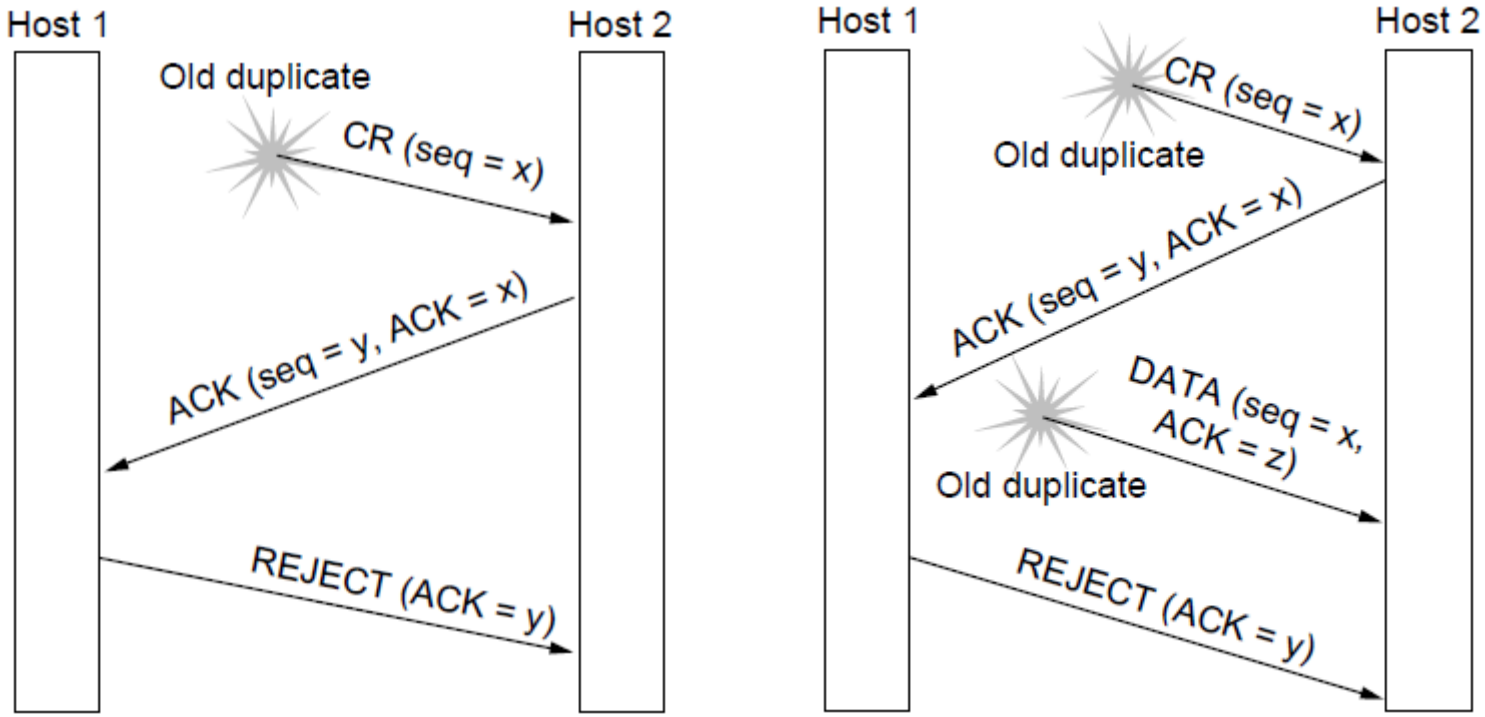
连接建立(3)

- 用序号区分重复的段！
- 建立连接时，接收方可以知道发送方的初始序号；如何让发送方知道接收方的初始序号？
  ——三次握手！
- 主机1发送CR(Connect Request)，初始序号为X
- 主机2发送ACK，确认主机1的序号X，并通告其初始序号为Y
- 主机1发送DATA，初始序号为X，并确认主机2的序号Y

三次握手保证双方序号同步



连接建立(4)



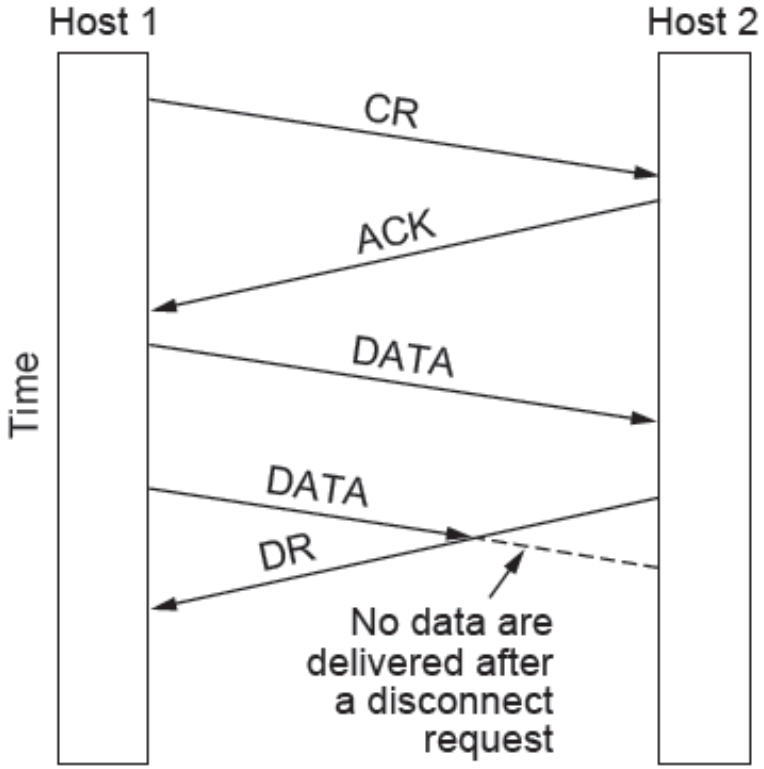
- 重复CR情况：主机2收到序号为X重复的CR并响应ACK，主机1根据其序号可判定是延迟的CR，则拒绝连接
- 重复CR和ACK情况：若主机2收到DATA，根据其ACK序号z而不是y可知DATA为重复的

连接建立(5)

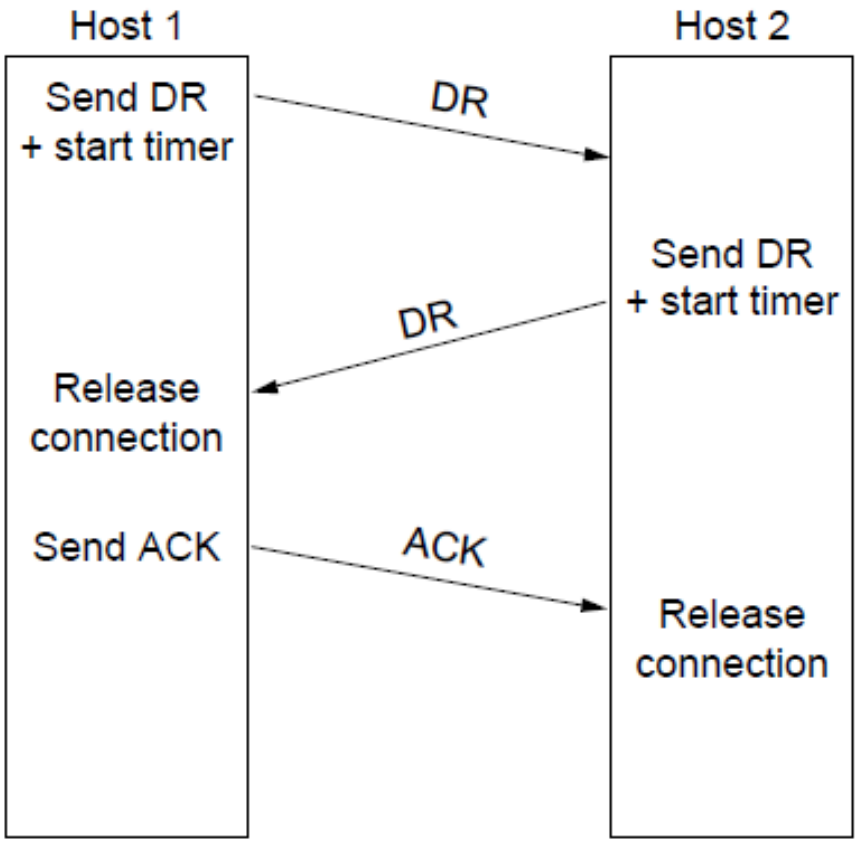
- TCP最初方案是基于系统时钟设置初始序号，并采用32位序号防止序号回绕（PAWS： Protection Against Wrapped Sequence Numbers）
- 基于时间戳设置初始序号，攻击者可以基于时钟预测下一个初始序号然后发送数据欺骗三次握手，建立一个伪造连接
- 现有的TCP连接采用伪随机初始序号；同样需要保证在一定时间（生存期）内序号不能重复

连接释放(1)

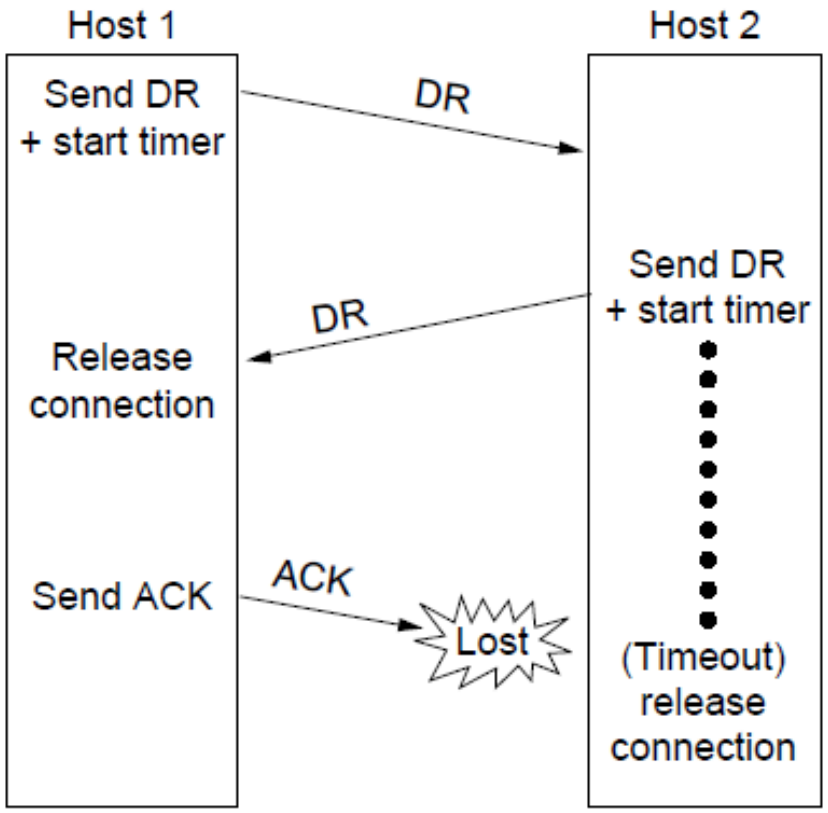
- 主机2突然终止连接，导致主机1无法传输数据
- 连接释放：采用安全释放连接方式，任何一方发送连接释放请求DR CR(Disconnect Request），都要等待另一方确认



连接释放(3)



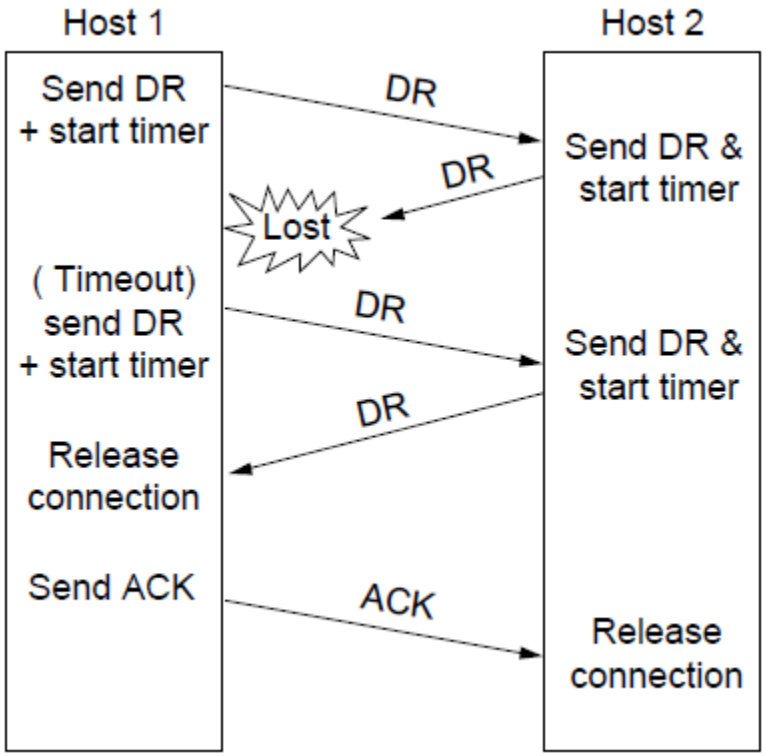
正常的三次挥手情况

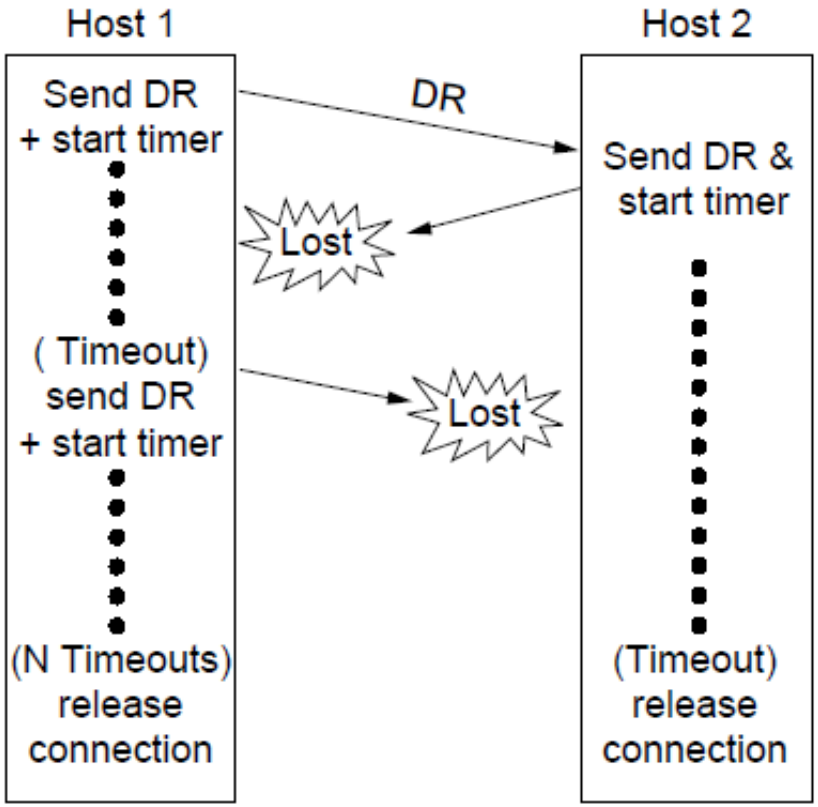


最后的ACK丢失

主机2发送DR并启动定时器，若定时器到则释放连接

连接释放(4)





响应丢失了　　响应及之后的DR都丢失了，致使主机1无法关闭连接！

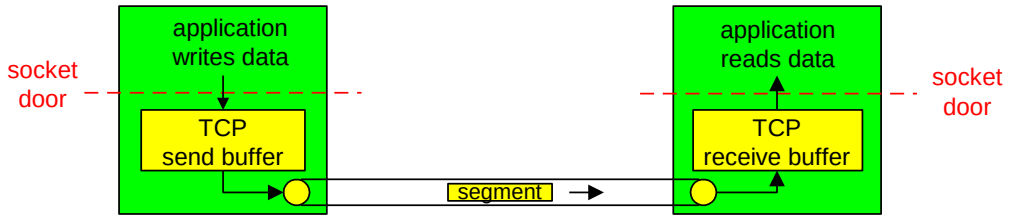
杀死半连接：一定时间内没有数据到来，则自动断开

提纲

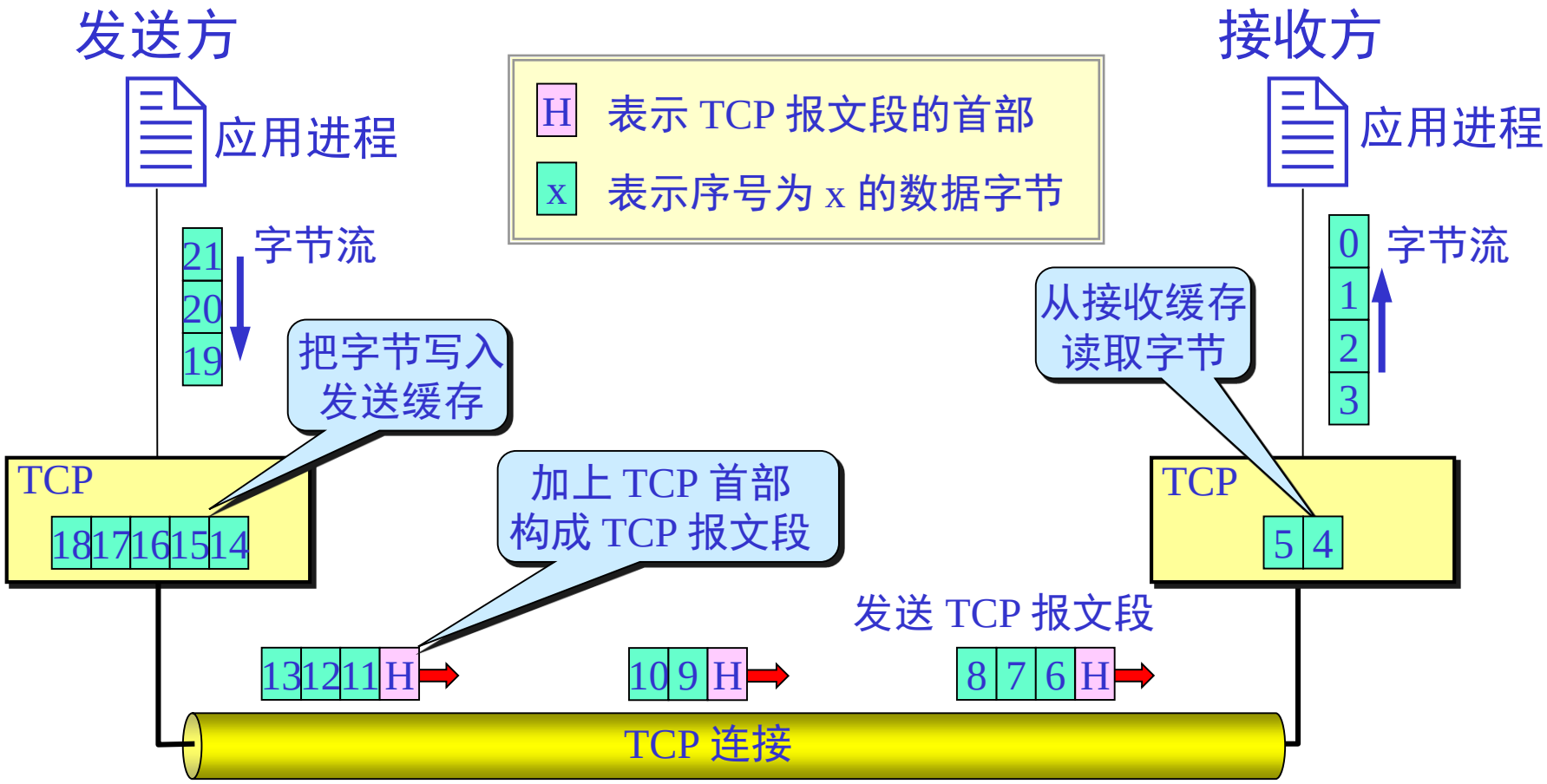
- 可靠传输机制
  - 超时重传
  - 肯定确认
- 面向连接的连接管理
  - 二次握手
  - 三次握手
- TCP协议
  - 三次握手
  - 滑动窗口与流量控制

TCP： RFC793, 1122, 1323, 2018, 2581

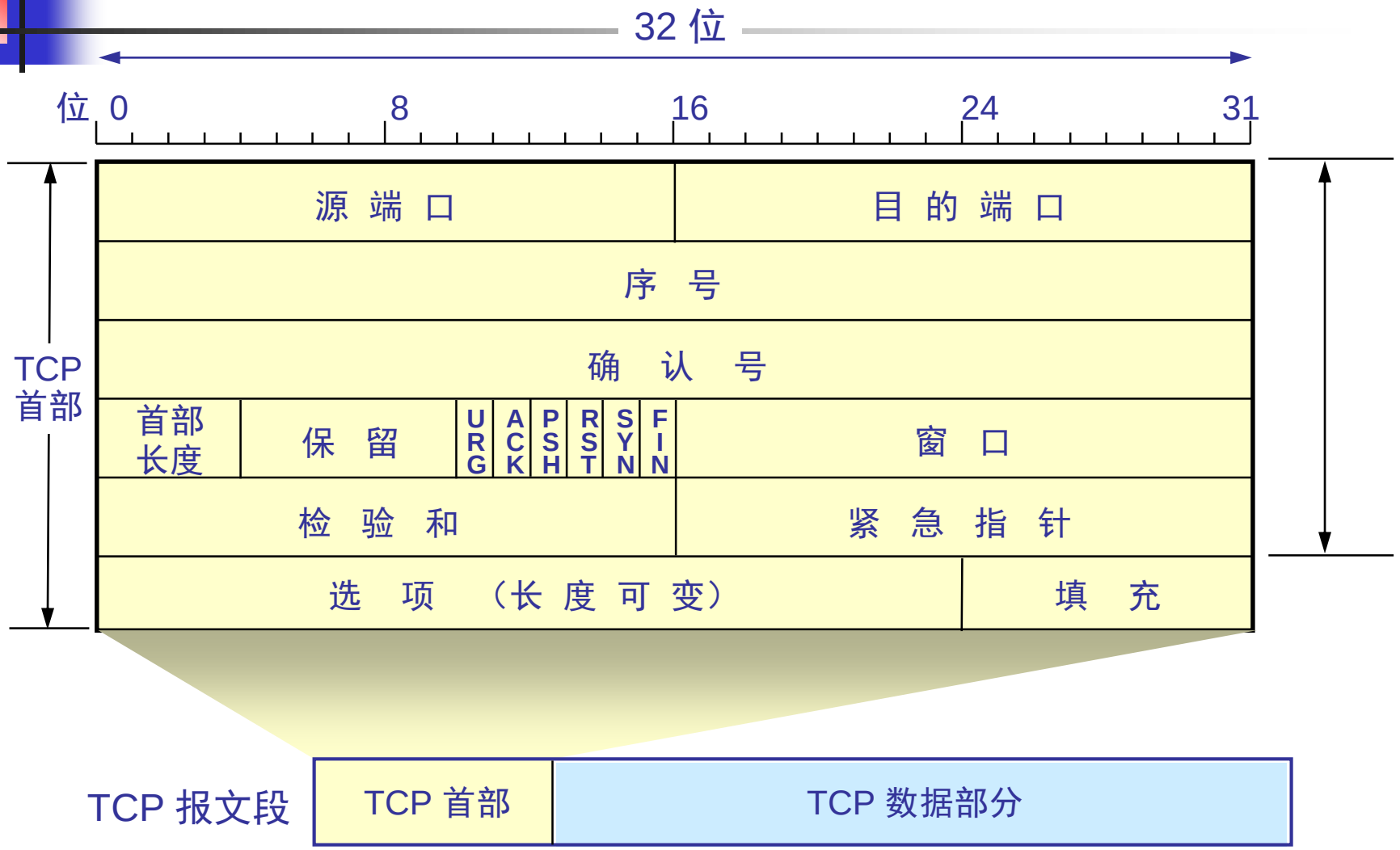
- 端到端
  - 一个发端，一个收端
- 可靠，按序的字节流
  - 无“数据流边界”
- 管道
  - TCP 拥塞及流量控制，设置窗口值
- 发送缓存与接收缓存
- 全双工数据传输
  - 一条连接支持双向数据流
  - MSS: maximum segment size，最大报文段长度
- 面向连接
  - 在数据发送前握手，初始化收发端状态
- 流量控制
  - 收端控制发端



传输控制协议 TCP：面向流的概念



TCP 报文段的格式


32 位
位 0
8
16
24
31
源 端 口
目 的 端 口
序 号
确 认 号
首部长度
保 留
URG
ACK
PSH
RST
SYN
FIN
窗 口
检 验 和
紧 急 指 针
选 项 （长 度 可 变）
填 充
TCP 首部
TCP 报文段
TCP 首部
TCP 数据部分

TCP首部格式

- 序号：在报文段中发送数据中第1个字节的序号
- 确认号：是期望收到的后续报文段数据的第一个字节的序号
- 首部长度：TCP 报文段首部长度，单位是4字节
- URG：置1，报文段为紧急数据
- ACK：置1，确认号有效
- PSH：置1，尽快交付，不再等到缓存满后再交付
- RST：置1，连接差错，重新建立连接
- SYN：置1，连接请求或连接接受
- FIN：置1，释放连接
- 窗口：收方设置发方的发送窗口，单位为字节
- 检验和：检验范围包括首部、数据和伪首部（类似于UDP）
- 紧急指针：紧急数据的字节数（紧急数据在报文段的最前面）

TCP首部格式——选项

- 选项最多为40字节，为4的倍数
- MSS：最大报文段数，增大报文，提高传输效率，默认为536，TCP为536+20
- 更大的窗口，最多为16+14位。因64KB窗口，在大的时延带宽积的信道上效率低，应设置更大的窗口
- 时间戳：用于计算RTT，根据时间戳丢弃接收的报文段，防止序号回绕
- SACK：告知发送方已经接收的序号范围

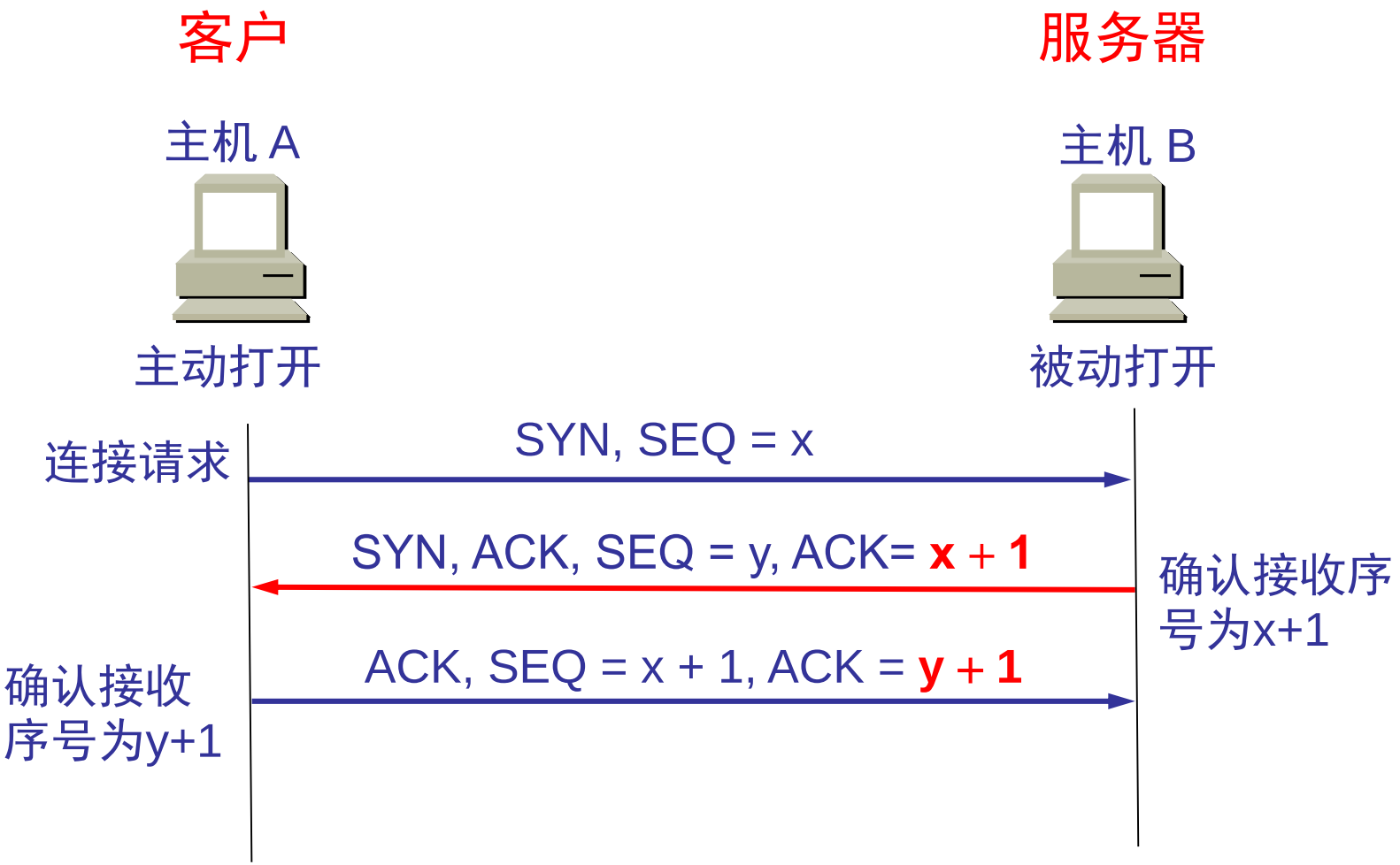
TCP的可靠传输

- 差错控制：ARQ
  - 对报文进行编号
  - 发送数据后并缓存，启动超时定时器，等待确认
  - 无确认则自动重传缓存的数据
- 流量控制：
  - 以字节为单位的滑动窗口
  - 序号占32位，为何如此之大？
- 连接建立
  - 三次握手（为何不是二次？）

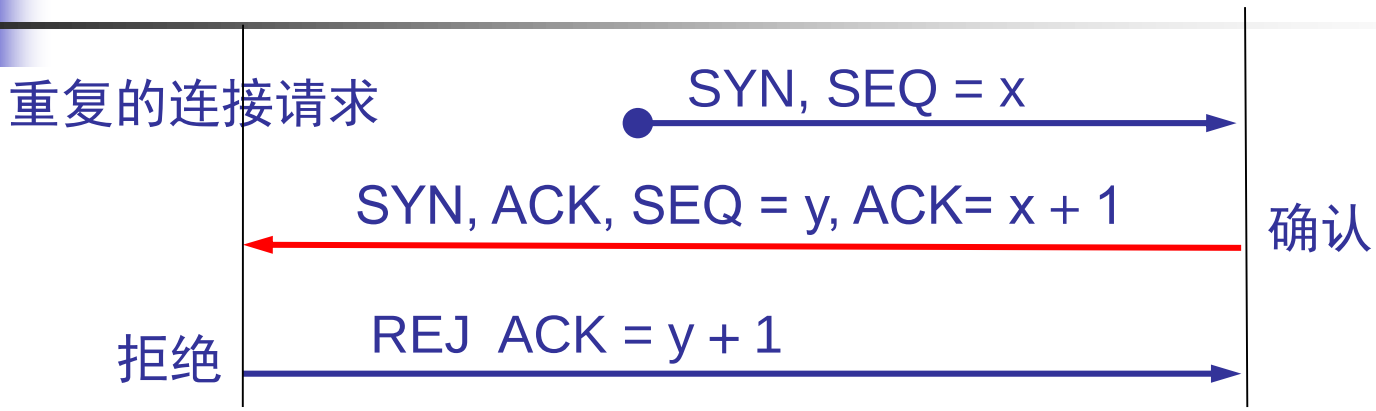
TCP 的连接管理

- TCP连接的三个阶段：连接建立、数据传送和连接释放
- 连接建立过程中要解决的问题：
  - 使每一方能够确知对方的存在
  - 允许双方协商一些参数（如最大报文段长度，最大窗口大小，服务质量等）
  - 对传输实体分配资源（如缓存大小，连接表中的项目等）
- 连接采用C/S方式：主动发起连接建立的进程为客户(client)，被动等待连接建立的进程为服务器(server)

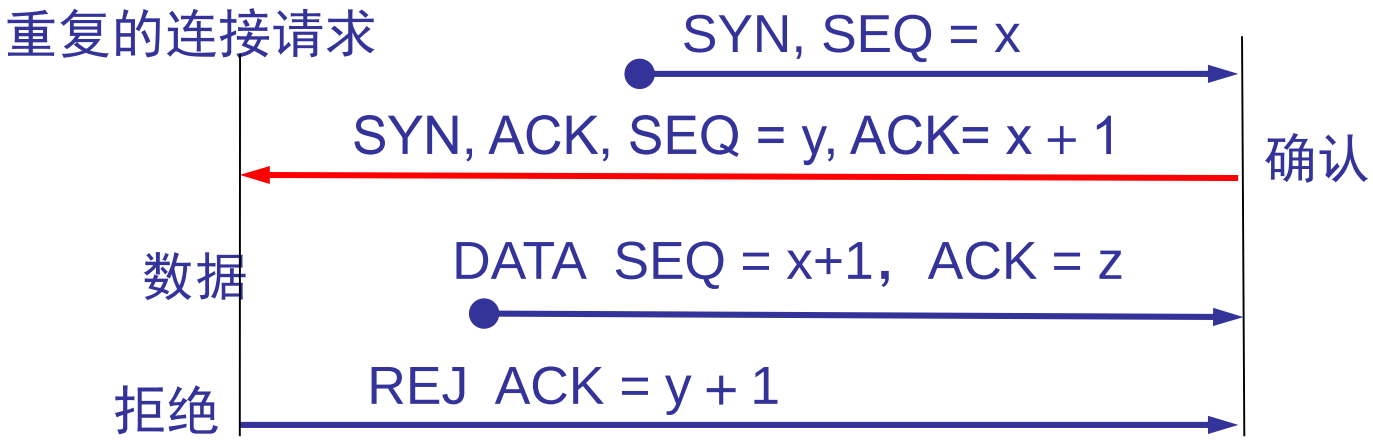
TCP 连接建立：三次握手



TCP连接建立：两端序号协调失败



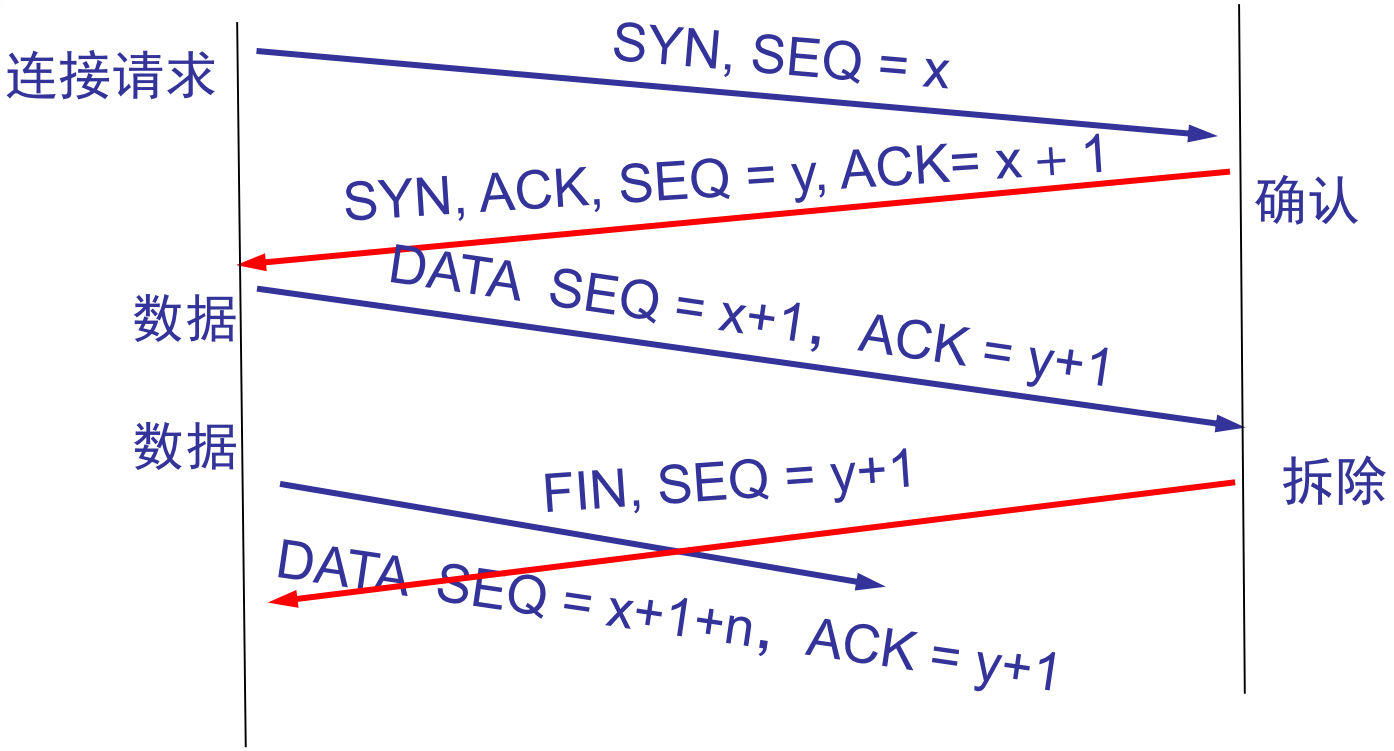
主机A的发送序号>x+1，当出现旧的连接请求



主机A的发送序号>x+1，主机B的发送序号为y+1

出现旧的连接请求及数据

连接释放

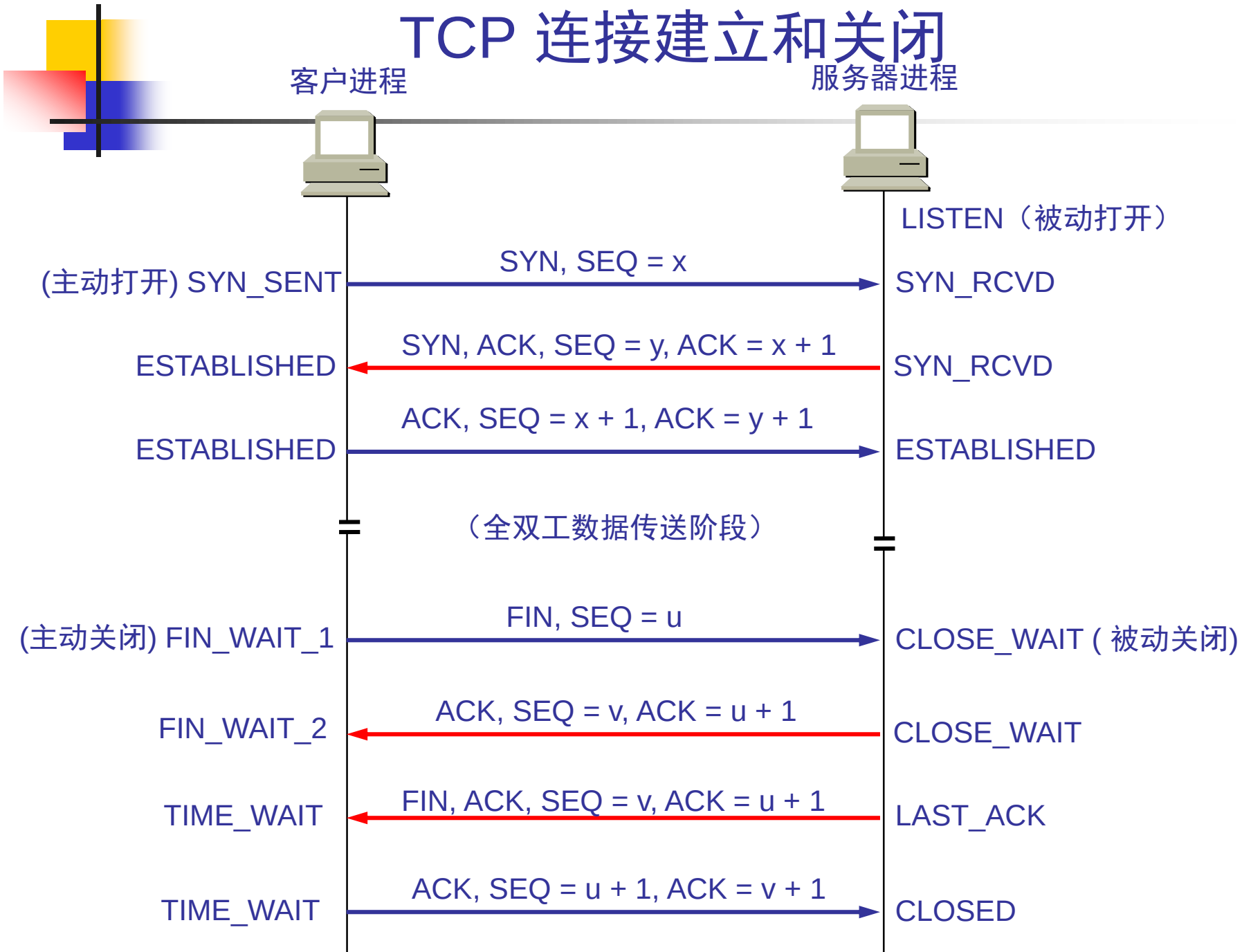


突然终止连接导致数据丢失

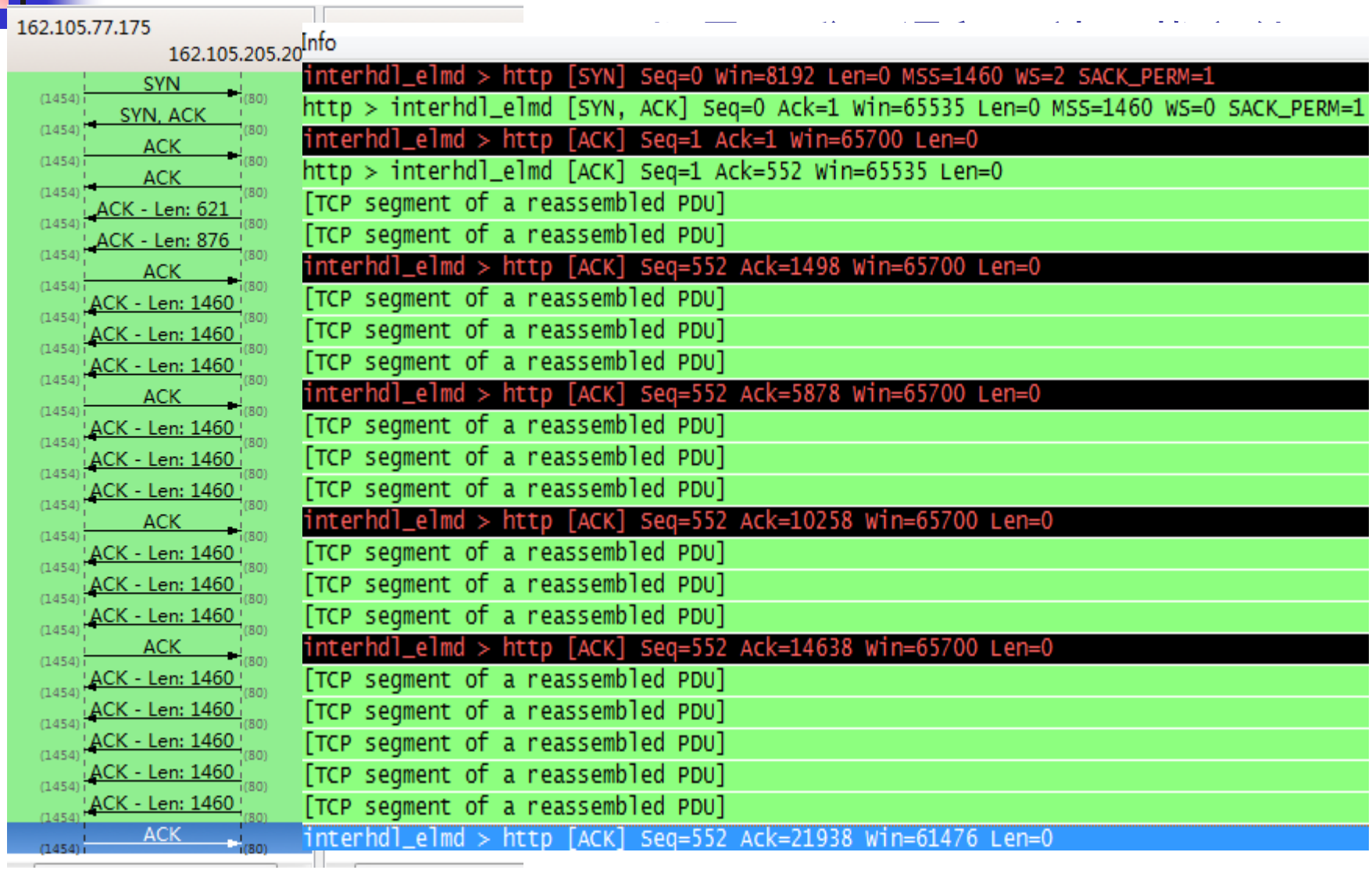
TCP 连接释放过程

至此，整个连接已经全部释放

TCP 连接建立和关闭


客户进程
服务器进程
LISTEN（被动打开）
(主动打开) SYN_SENT
SYN, SEQ = x
SYN_RCVD
ESTABLISHED
SYN, ACK, SEQ = y, ACK = x + 1
SYN_RCVD
ESTABLISHED
ACK, SEQ = x + 1, ACK = y + 1
ESTABLISHED
（全双工数据传送阶段）
(主动关闭) FIN_WAIT_1
FIN, SEQ = u
CLOSE_WAIT (被动关闭)
FIN_WAIT_2
ACK, SEQ = v, ACK = u + 1
CLOSE_WAIT
TIME_WAIT
FIN, ACK, SEQ = v, ACK = u + 1
LAST_ACK
TIME_WAIT
ACK, SEQ = u + 1, ACK = v + 1
CLOSED

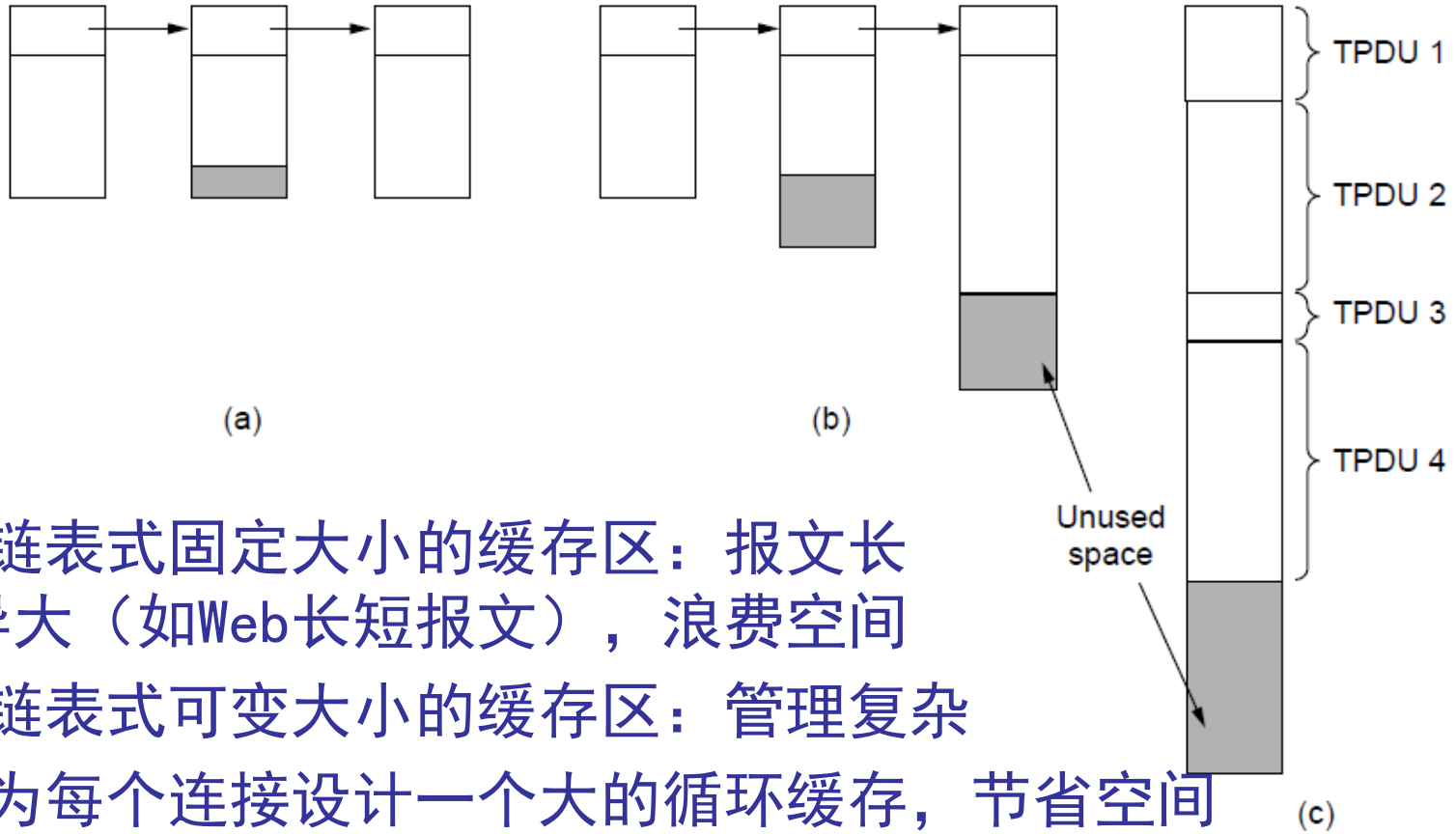
TCP：建立连接与数据传输示例



提纲

- 可靠传输机制
  - 超时重传
  - 肯定确认
- 面向连接的连接管理
  - 二次握手
  - 三次握手
- TCP协议
  - 三次握手
  - 滑动窗口与流量控制

滑动窗口与流量控制(1)



（a）链表式固定大小的缓存区：报文长度差异大（如Web长短报文），浪费空间

（b）链表式可变大小的缓存区：管理复杂

（c）为每个连接设计一个大的循环缓存，节省空间

窗口及流量控制是基于字节流的，每个报文段的长度也可变

滑动窗口与流量控制(2)

	A	Message	B	Comments
1	→	< request 8 buffers>	→	A需要8个缓存
2	←	<ack = 15, buf = 4>	←	B只同意接收4个消息
3	→	<seq = 0, data = m0>	→	A发送消息0，还剩3个缓存
4	→	<seq = 1, data = m1>	→	A发送消息1，还剩2个缓存
5	→	<seq = 2, data = m2>	•••	A发送消息2，还剩1个缓存，但消息丢了
6	←	<ack = 1, buf = 3>	←	B确认0-1，可接收3个(2～4)消息
7	→	<seq = 3, data = m3>	→	A发送消息3，有1个缓存
8	→	<seq = 4, data = m4>	→	A发送消息4，缓存耗尽
9	→	<seq = 2, data = m2>	→	A超时并重传消息2
10	←	<ack = 4, buf = 0>	←	**B确认所有消息，缓存为0，A被阻塞**
11	←	<ack = 4, buf = 1>	←	A可以发送消息5
12	←	<ack = 4, buf = 2>	←	B新找到1个缓存
13	→	<seq = 5, data = m5>	→	A剩1个缓存
14	→	<seq = 6, data = m6>	→	A再次被阻塞
15	←	<ack = 6, buf = 0>	←	**确认消息6，A被阻塞**
16	•••	<ack = 6, buf = 4>	←	消息丢失，A记录的缓存为0，死锁

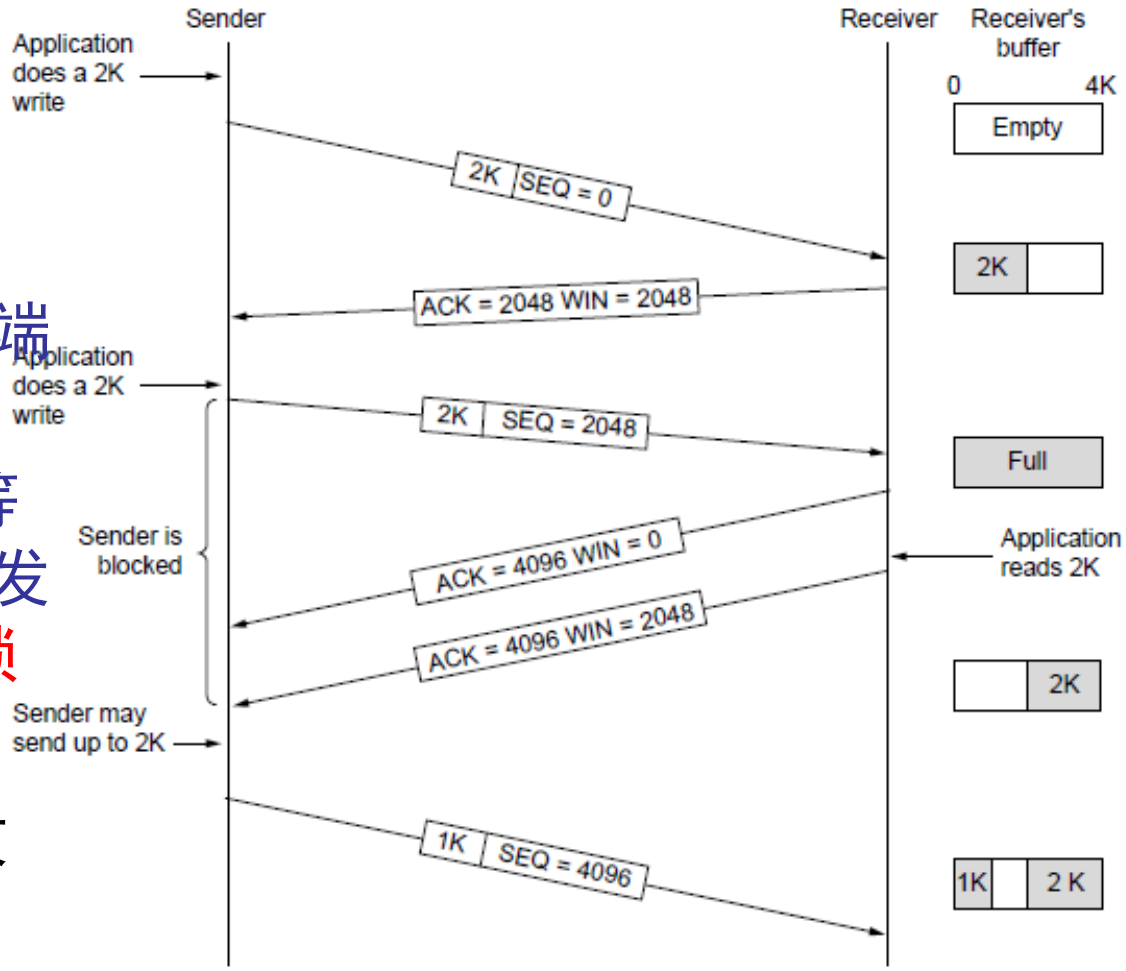
动态缓存区的分配场景

箭头表示传输方向，. . . 表示丢失报文段TPDU

B发送响应：收到新报文或缓存区变化

TCP滑动窗口与流量控制(1)

- 接收窗口为4KB
- 发送2KB
- 再发送2KB
- 当接收窗口为0时，发端被阻塞
- 在阻塞情况下，发端等待接收窗口不为0或者发送如下信息以避免死锁
  - 紧急数据
  - 1字节的段，强制收端通告其窗口



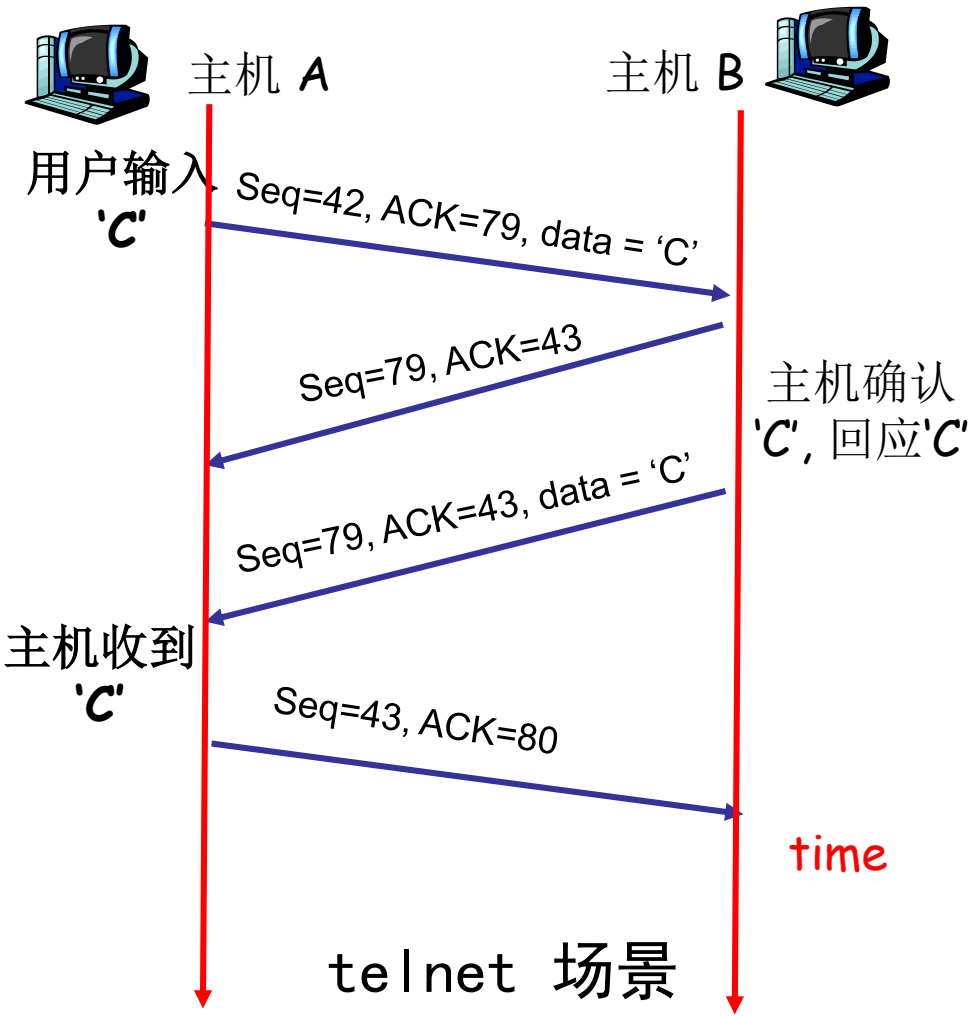
TCP的窗口管理

TCP滑动窗口

序号：报文段中首字节在流中的"序号"

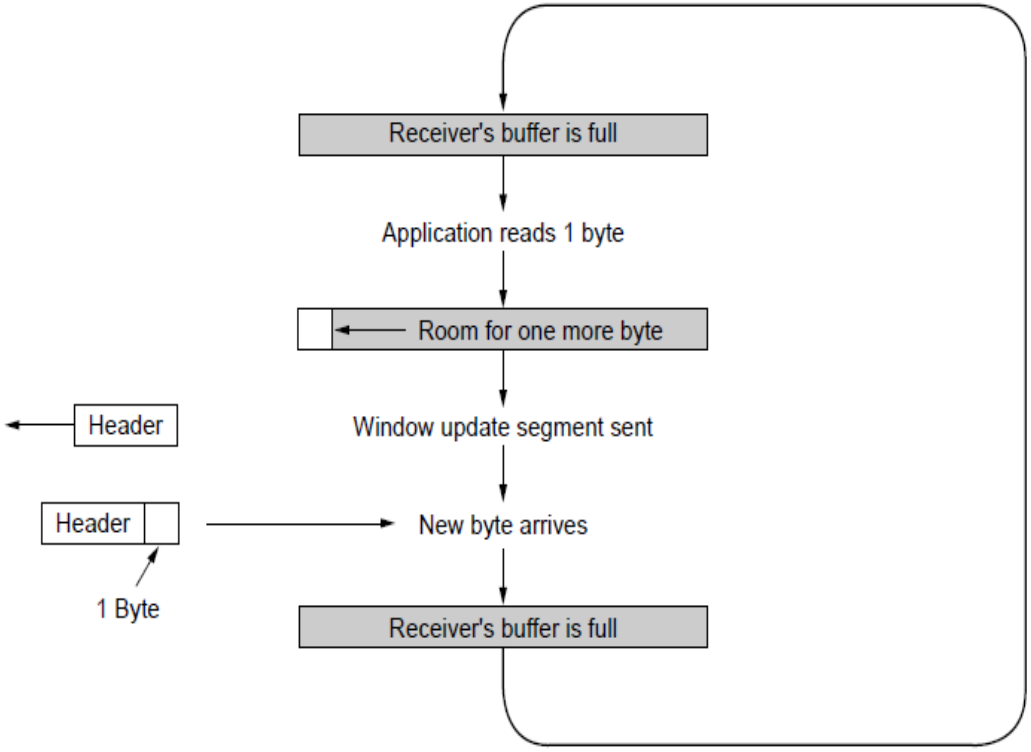
ACKs：期望接收的下一字节序号

- 远程终端连接问题
  - 发送1字符（TCP报文长41B），确认(报文长40B)，回显字符（报文长41B），确认(报文长40B)；传输1个字符需要交互4个报文共162字节！
  - 改进：接收端延迟50ms后确认并更新窗口，采用捎带确认；则传输1个字符，实际交互41×2字节！



TCP 滑动窗口(2)

- 当接收端收到大数据块，缓存区满，之后每次读1字节后就通告其窗口为1——低效窗口问题！
- Clark方案：禁止收端通告1字节的窗口更新段，仅当窗口为建立连接时所通告的最大数据段或其缓存的1/2时，才发送窗口更新段
- 提高效率的方法：发端不发送太短的数据段，收端不通告太小的接收窗口

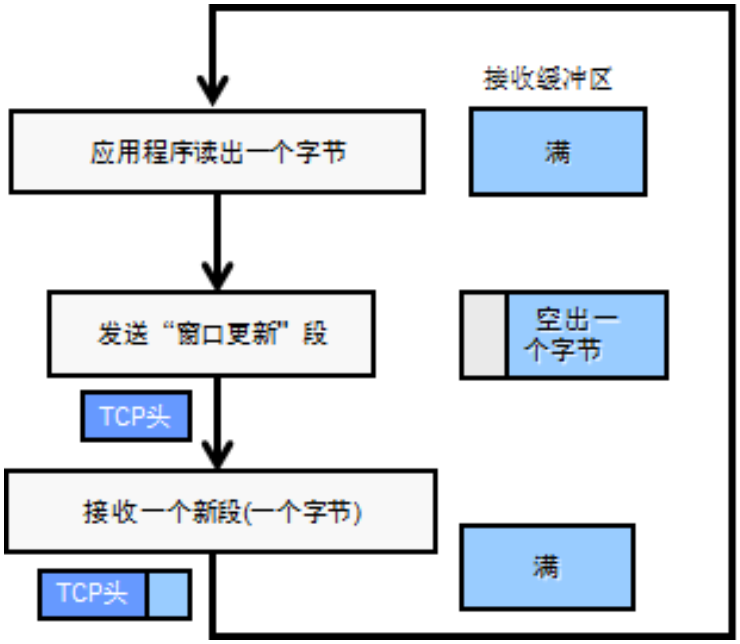


小结

- 在数据报网络之上，TCP提供可靠传输
- 连接建立需三次握手，实现收发双方的序号同步
- 基于字节流的滑动窗口与流量控制：
  - 窗口及流量控制是基于字节流的，一个连接一个大缓存，报文长度可变，灵活且有效
  - 序号占32位，避免序号回绕
  - 提高效率：发端不发送太小的报文段，收端不通告太小的接收窗口
- 遗留问题：拥塞控制、超时重传

练习题

- 在TCP传输中可能出现如右图的情形：接收端的缓冲区以及向发送端反馈的信息。试分析这种情况下的传输性能，并给出Clark解决方案
-
-



练习题

- 考虑A和B刚建立TCP连接，准备发送数据。假设A的初始序号为x，B端的初始序号为y。B暂时没有数据需要发给A，并且B的接收缓冲区为1000字节。

发送端A的发送序列

0,1, ……,99	100, ……,199	200,……,299	300,……,399	400,……,499

- a)针对A的发送序列，在网络情况良好下，试问B收到每个报文后给A反馈的确认帧的序号（seq）、确认号（ack）以及接收窗口分别是多少？

接收端B的接收序列

0,1, ……,99	出错，被丢弃	200,……,299	300,……,399	400,……,499

- b)若B的接收序列如图所示。试问B收到每个报文后向A反馈的确认帧的序号（seq）、确认号（ack）以及接收窗口分别是多少？

作业：TCP协议实验

- 1) 设计保存TCP 连接信息的数据结构（TCB）
- 2) 实现stud_tcp_input()函数进行TCP的接收：完成校验和检查、字节序转换，实现TCP客户机报文接收的有限状态机。
- 3) 实现stud_tcp_output()函数进行TCP的发送。
- 4)实现TCP客户机的5个Socket接口函数，stud_tcp_socket()、stud_tcp_connect()、stud_tcp_recv()、stud_tcp_send()和stud_tcp_close()，并与TCP发送和接收流程有机地结合起来

提交截止时间：5月30日