



## 第六章 传输层协议

### 课前思考

- 传输层的任务是什么？
- 传输层协议主要涉及哪些内容？
- 为什么说传输层协议是真正的端到端协议？
- 为什么传输连接需要三次握手？
- TCP与UDP区别是什么？
- 传输地址与网络地址有什么不同？
- 出错或丢失的IP分组会重发吗？





# 本章内容

---

- 6.1 传输层概述**
- 6.2 传输服务质量**
- 6.3 传输层协议机制**
- 6.4 TCP协议**
- 6.5 UDP协议**



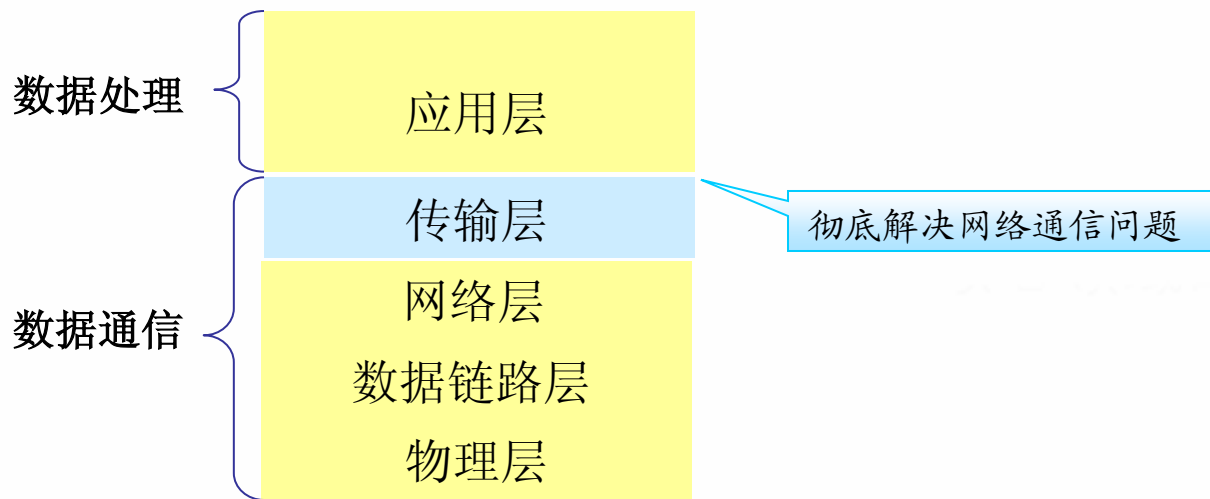
## 6.1 传输层概述

- 传输层功能

完成主机进程—主机进程之间的报文传输。

- 传输层地位

位于网络体系结构的中间，作为数据通信和数据处理的分水岭，具有承上启下的作用。





## 6.1 传输层概述

- 传输层要彻底解决网络通信问题，使得应用层不必关心通信问题。
- 传输层是真正的端对端的通信。
  - 传输层协议在端主机上运行，路由器一般没有传输层。
  - 传输层从主机层面上对网络层采取相应补救措施，可以提供更高质量的数据传输能力。
  - 传输层独立于网络设备，可以提供标准的传输服务接口（原语）。
- 传输层协议涉及的内容
  - 寻址：传输地址及其发现。
  - 差错控制：分组顺序、差错检测、重传确认机制等。
  - 连接管理：连接建立、连接释放。
  - 流量控制：主机进程之间的流量控制。



## 6.2 传输服务质量 (QoS)

- 传输服务质量反映网络最终通信质量

- 传输服务质量参数

- 连接建立延迟

发出连接请求到连接建立成功之间的时间。

- 连接建立失败概率

- 吞吐量

在一条传输连接上，每秒传输有效数据的字节数。

- 传输延迟

从开始传输数据到该数据被收到为止的时间。

- 残留差错率

传输连接上数据出错的概率。



## 6.2 传输服务质量 (QoS)

- 保护性

传输过程中保障数据安全的能力，包括防止非法窃取、篡改数据等。

- 优先权

保证高优先权的传输连接优先传输数据、优先获得网络资源的能力。

- 回弹率

由于某种原因而自发终止传输连接的可能性。



## 6.3 传输层协议机制

### 6.3.1 传输协议数据单元

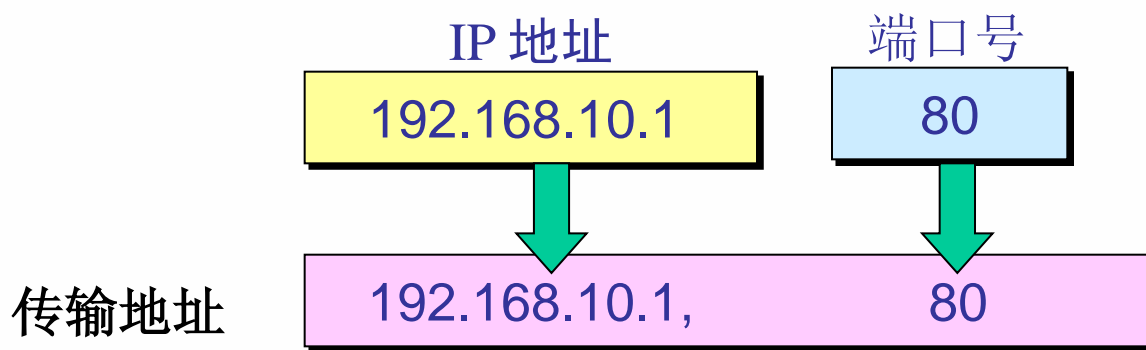
- 传输协议数据单元（TPDU）是传输层的最小数据传输单位，又称为“报文段”。
- TPDU类型
  - CR: 连接请求
  - CC: 连接确认，对CR TPDU的确认
  - DR: 断连请求
  - DC: 断连确认，对DR TPDU的确认
  - DT: 数据（高层数据）
  - AK: 数据确认，对DT TPDU的确认
  - REJ: 拒绝接受请求，或拒绝接受DT TPDU



## 6.3 传输层协议机制

### 6.3.2 传输地址

- 传输地址唯一地标识主机进程
- 传输地址=网络号+主机号+端口号，端口号即传输服务访问点（TSAP），用来标识应用进程。
- 在IP网络，传输地址= IP地址+端口号



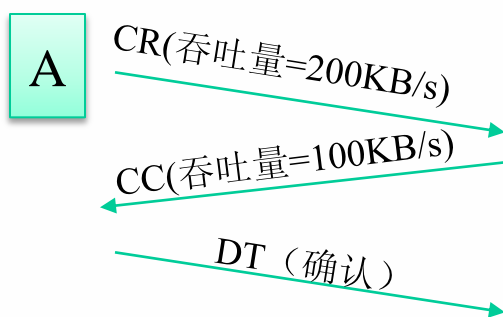




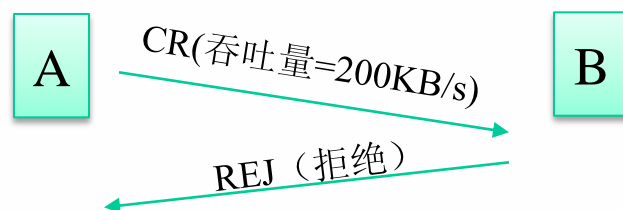
## 6.3 传输层协议机制

### 6.3.3 传输连接

- 传输连接实质上是通过传输地址,建立两个主机进程之间的连接。
- 其他层上的连接一般采用“**两次握手**”,建立传输连接过程比较复杂,双方需进行反复协商,通常采用“**三次握手**”,其原因是:
  - 传输服务质量等参数进行多次协商,有助于提高连接成功率。



三次握手连接成功



二次握手连接失败

- 网络层服务不可靠 (如IP协议), “**两次握手**”可能导致连接失败。

P.267: 图6.10, 图6.11



## 6.3 传输层协议机制

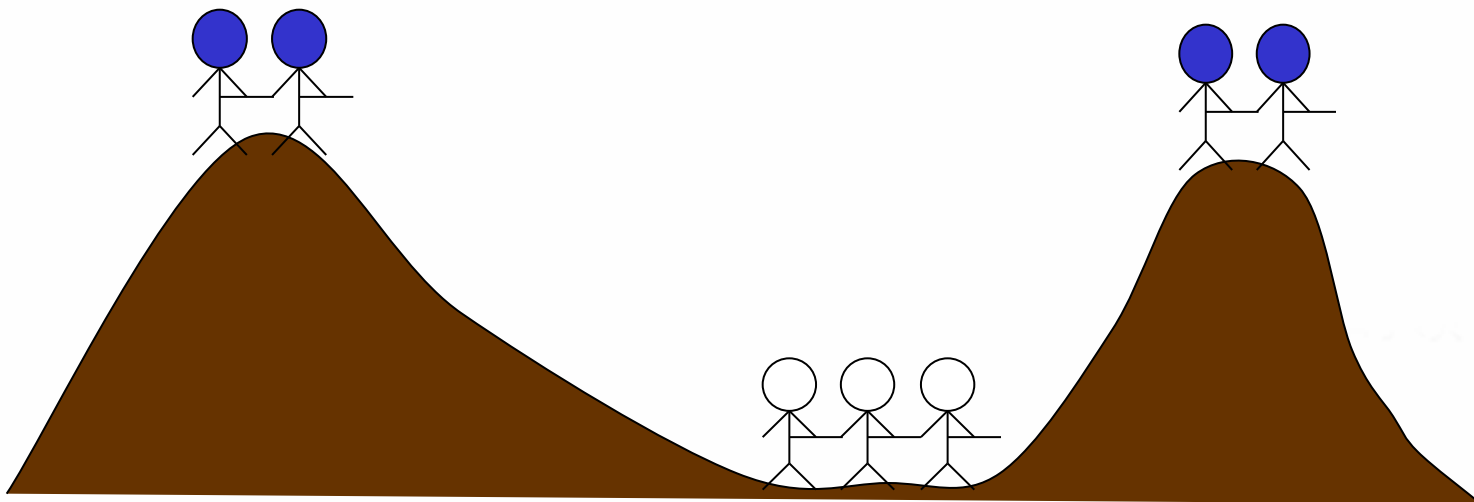
- 建立传输连接过程
  - 发起方发出建立连接请求CR TPDU，提出相关服务质量要求及其它参数与对方协商，包括：
    - 吞吐量
    - TPDU的大小等
    - 传输层是否进行流量和差错控制
    - 是否允许加密数据
    - 最长等待时间（确认时间）
    - .....
  - 接收方收到连接请求CR TPDU后，
    - 若同意（或适当修改协商参数），返回连接确认CC TPDU。
    - 若不同意，则返回断连请求DR TPDU，并附带原因参数。
  - 发送方收到连接确认CC TPDU后，
    - 发送第一个数据TPDU（DT TPDU）确认连接建立；
    - 发送REJ TPDU拒绝连接。



## 6.3 传输层协议机制

- 释放传输连接
  - 白军/蓝军问题

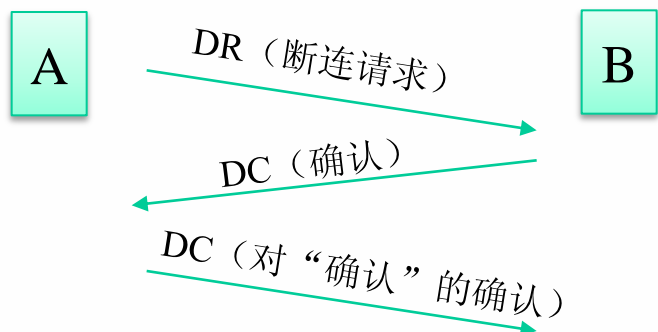
占据两个山顶的蓝军包围了山谷中的白军。两个山顶上的蓝军同时出击，则白军必败；若一个山顶的蓝军单独出击，则蓝军必败。所以，两个山顶上的蓝军若要取胜，则必须派出通讯员，穿过山谷（可能被白军俘虏），约定同时出击。你能设计出这种通信协议吗？





## 6.3 传输层协议机制

- 释放连接仍采用“三次握手”



三次握手释放连接



## 6.4 TCP协议

### 6.4.1 TCP概述

- Internet的两个传输层协议
  - TCP (Transmission Control Protocol) : 在IP支持下, 为应用层提供面向连接的、可靠的端到端字节流服务。
  - UDP (User Datagram Protocol) : 为应用层提供非面向连接的、不可靠的传输服务。
- TCP传输地址 (套接字) : IP地址+端口号
  - TCP连接用四元组<源IP地址, 源端口号, 目的IP地址, 目的端口号>表示。
  - 端口号16位, 端口号范围: 0 - 65535 ; 其中0-1023称为著名端口号, 分配标准服务使用, 如FTP服务器端口号为21, HTTP服务器端口号80, TELNET服务器端口号为23, SMTP服务器端口号为25。



## 6.4 TCP协议

- TCP最初在Unix环境下实现，后来也在Windows环境下实现，通过Socket提供服务，见P.261。
- 报文段（即TPDU）封装在IP分组中，IP分组封装在数据帧中。



- TCP提供全双工数据传输服务，如果主机进程A和主机进程B建立了一条传输链路，则意味着A，B都可以同时发送和接收TCP报文段。
- TCP不支持组播和广播。
- TCP连接提供可靠的字节流服务

如果发送了4个有效载荷为512B的报文段，接收方收到后，向应用层提供2048B的字节流，而不是4个报文段。这是因为TCP是真正的端到端通信，接收端无需再转发。



## 6.4 TCP协议

### 6.4.2 TCP报文段格式

TCP报文段是TCP协议的数据交换单元，其格式如下：





## 6.4 TCP协议

- **源/目的端口号**：各占16位，表示发送方和接收方的端口号。
- **顺序号**：32位，表示TCP段中的数据部分第1个字节的编号。
- **确认号**：32位，表示期望接收数据的第1个字节的编号，同时表示对确认号之前数据的确认。
- **数据偏移**：4位，表示TCP段头长度，包括固定和可变部分，单位为字（32位）
- **URG**：紧急数据标志。当有紧急数据时，该标志为“1”。
- **ACK**：该标志若为“1”，则表示**确认号**有效；若为“0”，则**确认号**无效。
- **PSH**：表示要求马上发送数据，不必等到缓冲区满时才发送。  
例如，TELNET协议中，每输入一个字符就必须立即发送。
- **RST**：该标志用于对本次TCP连接进行复位。通常在TCP连接发生故障时设置本位，以便双方重新同步，并初始化某些连接变量。如接收方收到它不希望接收的报文段，则将RST置“1”。





## 6.4 TCP协议

- **SYN:** 用于建立TCP连接。
  - SYN置为“1”且ACK置为“0”，表示请求建立TCP连接
  - SYN置为“1”且ACK置为“1”，表示确认TCP连接。
- **FIN:** 用于释放连接。若FIN置为“1”，则表示没有数据要发送了，但仍可以接收数据。
- **窗口大小:** 16位，用于TCP流量控制。表示从确认的字节号开始还可以接收多少字节。窗口大小也允许为“0”，表示确认号以前的字节已收到，但暂停接收数据。
- **校验和:** 用于对TCP报文段进行校验（类似于IP校验和），但校验范围包括头部和数据部分。计算校验和需包括一个TCP伪头，这样有助于检测分组是否被错误递交。

0	8	16	31
源IP地址			
目的IP地址			
0	协议（6）	TCP长度	

TCP伪头



## 6.4 TCP协议

- **紧急指针：**当URG为“1”时，紧急指针给出TCP段中紧急数据的长度，单位为字节；数据字段的起始位置（**顺序号**）作为紧急数据的开始位置。
- **选项：**目前TCP只规定一种选项，即最大的TCP段长（MSS），缺省值是556B。TCP实体双方可通过协商确定一个特定的最大TCP段长。

若MSS选的太小，则由于帧头和报头开销而网络利用率会降低；若MSS选的太大，则由于IP层分段而增加额外开销。一般认为，MSS应尽可能选大些，只要IP层不需要分段就行。
- **填充：**以0为填充位，确保TCP头部以32位边界结束。
- **数据：**TCP载荷数据，由于IP分组长度限制，TCP最大载荷长度=65535B-20B-20B=65495B。



## 6.4 TCP协议

### 6.4.3 TCP连接管理

- 建立TCP连接的目的

- 发现对方是否存在
- 进行参数协商（协议类、MSS、窗口大小、服务质量等）
- 为TCP实体分配资源（缓冲区、连接表等）

- TCP连接建立

- 采用C/S方式

- 建立TCP连接，首先要解决的问题是如何获得对方的端口号。

发起方C（客户机）首先发起连接请求。选择一个 $\geq 1024$ 的端口作为源端口，使用接收方S（服务器）的众所周知的端口号作为目的端口，如FTP使用21号端口，Telnet使用23号端口，SMTP使用25号端口，HTTP使用80号端口等。

- 接收方S（服务器）始终监听特定的端口，等待发起方C（客户端）的连接请求；当收到连接请求后，就知道了发起方C的端口号。

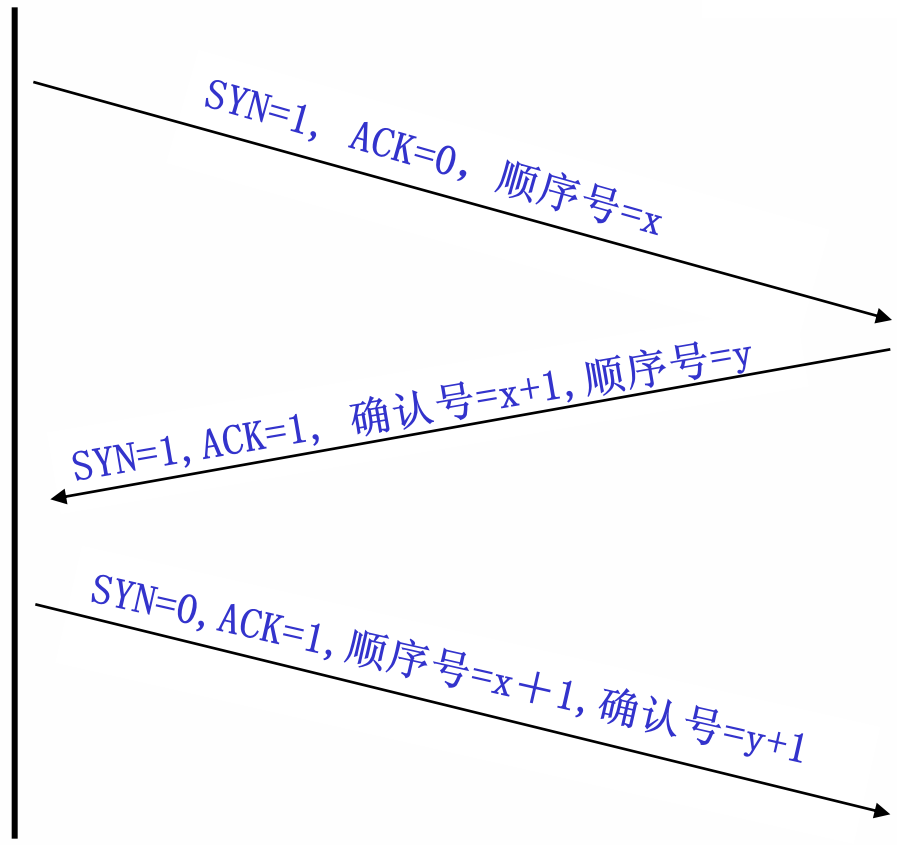


## 6.4 TCP协议

- 采用“三次握手”

客户机C（发起方）

服务器S（接收方）





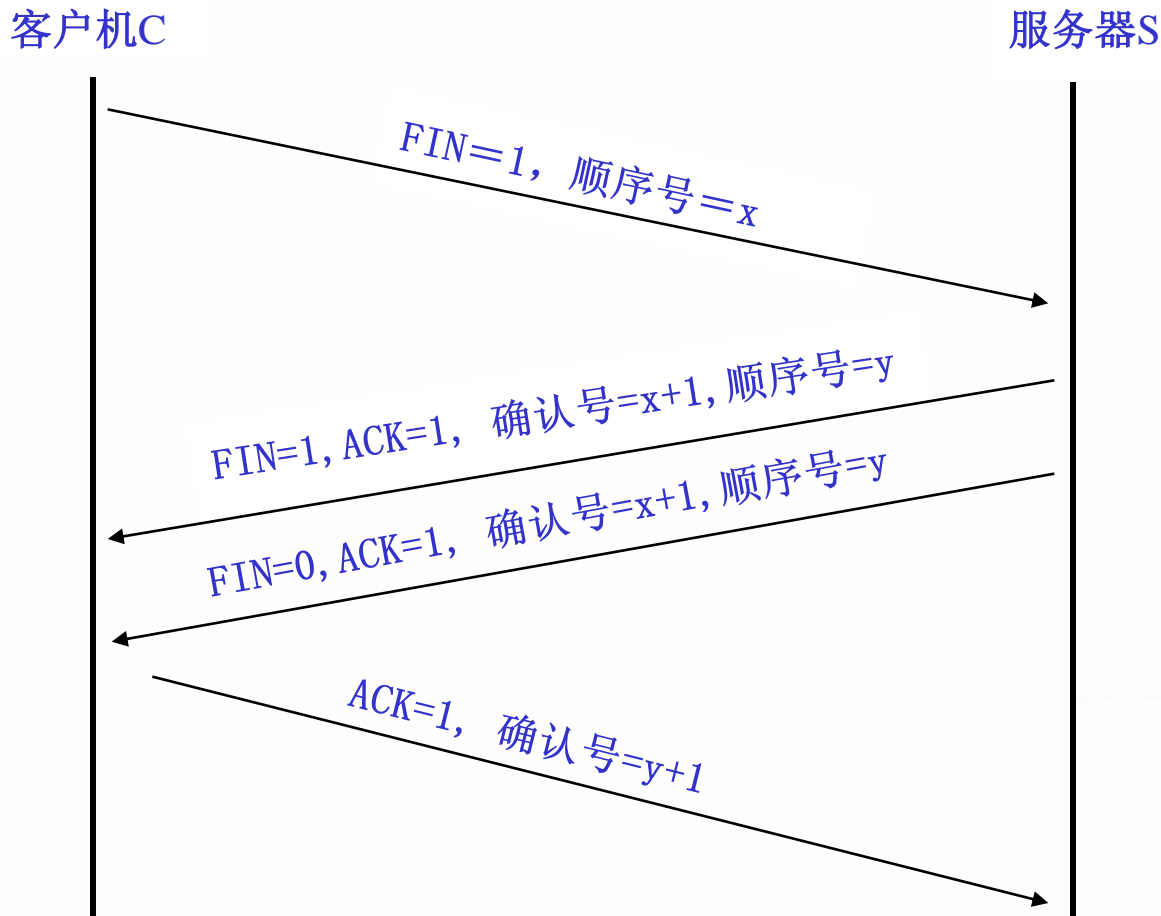
## 6.4 TCP协议

- 客户端发出连接请求TCP段 ( $\text{SYN}=1$ ,  $\text{ACK}=0$ , 顺序号= $x$ , 其中 $x$ 为随机数), 指明想要连接的服务器端口号 (目的端口), 设置TCP段的最大段长及其它参数。
- 服务器的TCP实体收到该请求后, 检查是否有进程在监听目的端口;
  - 如果没有, 则返回拒绝TCP段 ( $\text{RST}=1$ ) 的作为应答, 拒绝请求。
  - 如果有, 则该进程可以接受或拒绝连接请求:
    - 如果接受, 则返回确认TCP段 ( $\text{SYN}=1$ ,  $\text{ACK}=1$ , 顺序号= $y$ , 确认号= $x+1$ , 其中 $y$ 为随机数)。
    - 如果拒绝则返回拒绝TCP段 ( $\text{RST}=1$ )。
- 客户端收到确认TCP段后, 也发送一个确认TCP段 ( $\text{SYN}=0$ ,  $\text{ACK}=1$ ), 并允许该确认TCP段直接开始发送数据。



## 6.4 TCP协议

- TCP连接释放：仍采用“三次握手”





## 6.4 TCP协议

- 主机A发送断连请求TCP段（ $FIN=1$ ），向主机B表明“我已无数据要发送，但如果你要发送数据，我仍能接收”。
- 主机B收到断连请求后，
  - 如果无数据传输，则返回应答TCP段（ $FIN=1, ACK=1$ ）；
  - 如果有数据传输，则返回应答TCP段（ $FIN=0, ACK=1$ ）；
- 主机A返回确认TCP段（ $ACK=1$ ）。



## 6.4 TCP协议

### 6.4.4 TCP流量管理

- 为什么要进行流量控制

发送方TCP实体发送数据过快，接收方TCP实体来不及处理，导致接收缓冲区溢出。

- TCP流量控制采用可变尺寸的滑动窗口协议。

- 由于TCP为应用层提供字节流服务，窗口尺寸以字节为单位。
- 应用进程根据处理能力读取TCP接收缓冲区中的字节流，导致空闲的接收缓冲区（接收窗口）动态变化。

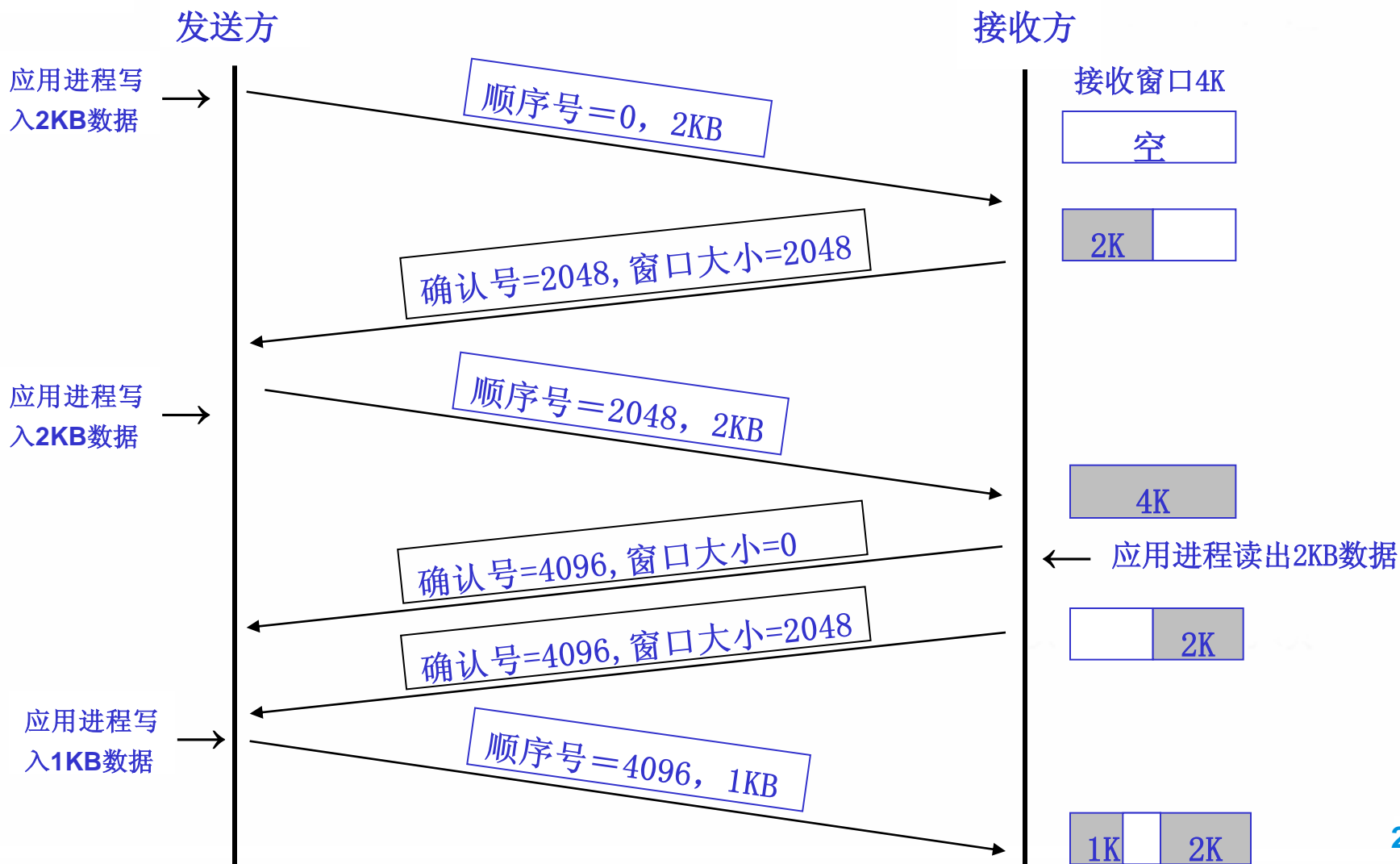
- TCP流量控制过程

类似于数据链路层的滑动窗口协议，所不同的是：通过接收窗口当前尺寸调整发送窗口的上限值。





## 6.4 TCP协议





## 6.4 TCP协议

### 6.4.5 TCP拥塞控制

- TCP拥塞控制与流量控制的区别
  - 拥塞控制是为了防止网络过载导致路由器缓冲区不足而造成IP分组的丢失。
  - 流量控制是为了防止发送端发送速度过快，导致接收端缓冲区不足而造成TCP段的丢失。
- 从理论上讲，网络过载（即网络拥塞）问题应由网络层（IP）解决，但IP没有实现拥塞控制功能，Internet拥塞控制实际上由TCP完成。
- TCP拥塞控制策略
  - 发送端设置**拥塞窗口**来反映网络容量，通过**拥塞窗口**来限制发送方向网络注入数据的速度，即发送端允许发送的数据既不能超过**接收窗口**的大小，也不能超过**拥塞窗口**的大小；前者是为了流量控制，后者是为了拥塞控制。
  - 通过**慢启动**和**拥塞避免策略**来控制拥塞窗口的大小。



## 6.4 TCP协议

### ● TCP拥塞控制过程

#### ● 慢启动

- 设：sssthresh为慢启动阈值，MSS为最大TCP段长度，cwnd为拥塞窗口大小。
- 发送端初始cwnd一般设置为1个MSS，sssthresh设置为接收窗口大小。
- 每收到一个确认ACK，则 $cwnd = cwnd + 1$ 个MSS。

初始 $cwnd=1$ ，发送端发送1个报文段 $M_0$ ，接收端收到后返回 $ACK_1$ 。发送端收到 $ACK_1$ 后，将 $cwnd$ 从1增大到2，于是接着发送 $M_1$ 和 $M_2$ 两个报文段。接收端收到后返回 $ACK_2$ 和 $ACK_3$ 。发送端收到后，将 $cwnd$ 从2增大到4，于是接着发送 $M_3 \sim M_6$ 共4个报文段。.....

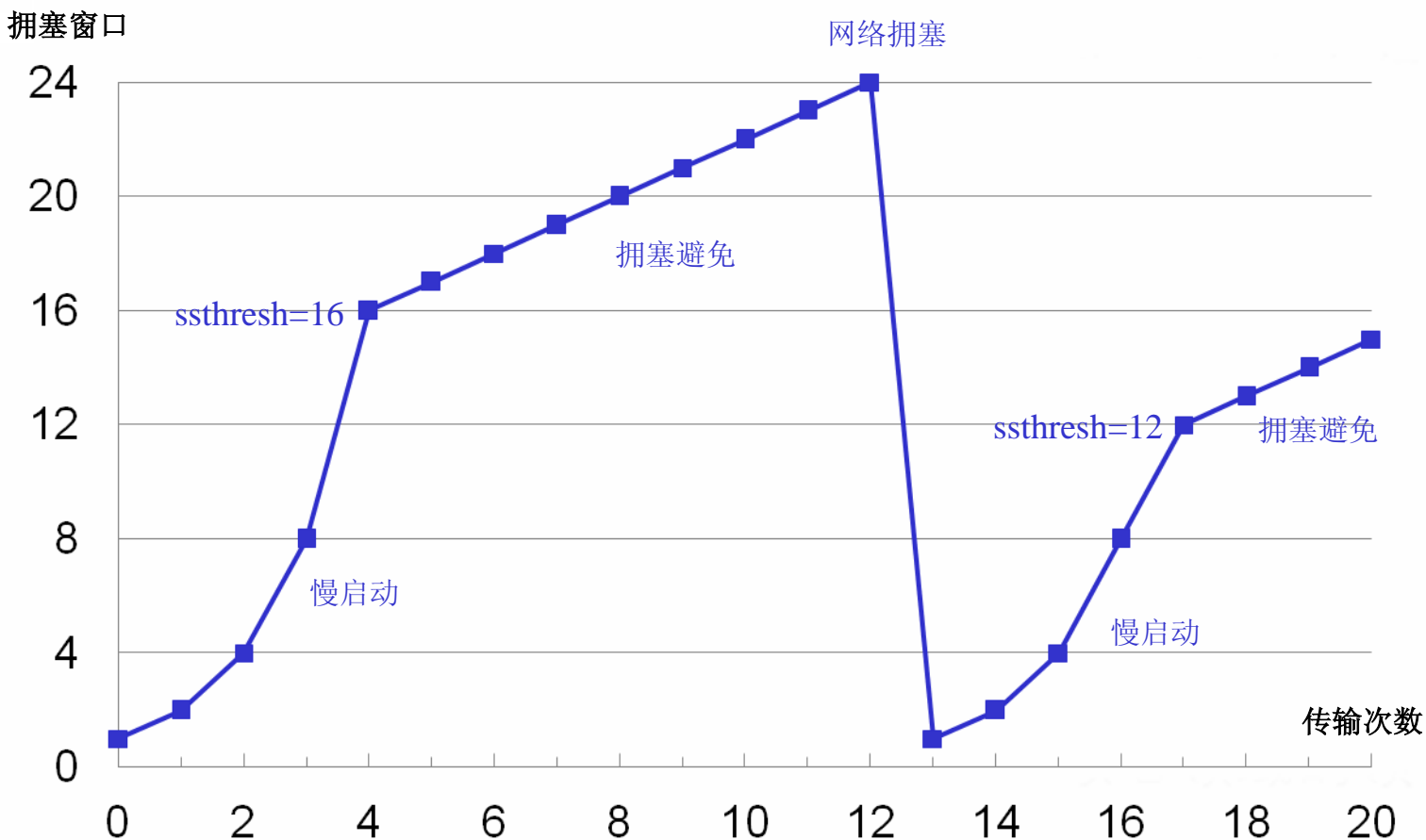
- 当 $cwnd > sssthresh$ 时，进入**拥塞避免**阶段。

#### ● 拥塞避免

- 每经过一个往返传输时间RTT（而不管在RTT 时间内收到几个ACK），则 $cwnd = cwnd + 1$ 个MSS；RTT是动态变化的。
- 如果超过一段时间（TCP重传超时）没有收到TCP报文段，则认为**网络拥塞**。
- 不管是慢启动阶段，还是拥塞避免阶段，如果TCP检测到拥塞，则将sssthresh缩减成 $cwnd$ 的一半，且 $cwnd$ 恢复为初始大小，即一个MSS。



## 6.4 TCP协议



- sssthresh初始值(16)为接收窗口大小。
- 慢启动不是指cwnd增长速度慢，而是指初始设置cwnd=1。



## 6.5 UDP协议

- UDP是非面向连接，不可靠的传输协议。
  - 不需要建立连接。
  - 不支持流量控制和拥塞控制。
  - UDP段在传输过程中可能会丢失、失序和延迟。
  - 支持广播和组播，这对多媒体传输是非常有用的。
- UDP实质上在IP基础上，增加了端口机制，实现了主机进程间的数据传输。
- UDP报文段

0	8	16	31
源端口号		目的端口号	
长度		检验和	
用户数据			



## 6.5 UDP协议

- UDP计算“校验和”方法同TCP，也需要包括一个UDP伪头；一旦发现出错，则简单地丢弃，“重传”由应用层完成。

0	8	16	31
源IP地址			
目的IP地址			
0	协议 (17)	UDP长度	

UDP伪头

- UDP采用**尽力而为**方式，即不保证可靠传输，既没有重传机制，也没有流量控制和拥塞控制，因此主机不需要维护具有许多参数、复杂的连接状态表，因此减少了开销和发送数据之前的延时，适合很多实时应用，如IP电话、视频会议等。



# 本章小结

## ● 内容

主要介绍传输层协议及其相关技术，包括传输服务质量（QoS）、传输地址、传输连接的建立和拆除、流量控制、拥塞控制、Internet的主要传输层协议实例TCP、UDP等。

## ● 重点

TCP协议原理，TCP报文段格式及每个字段的含义，传输连接的建立和拆除过程。