

# 计算机网络

## 第三章 传输层

---

谢瑞桃

[xie@szu.edu.cn](mailto:xie@szu.edu.cn)

[rtxie.github.io](https://github.com/rtxie)

计算机与软件学院

深圳大学



# 第三章讲解内容

## 1. 传输层概述与UDP

- 需求/服务/协议、多路复用/分解、UDP协议

## 2. 可靠传输

- 可靠传输基础知识、TCP可靠传输

## 3. TCP

- 报文段结构、超时间隔、流量控制、连接管理

## 4. TCP拥塞控制

- 网络拥塞、TCP拥塞控制、吞吐量分析



# 传输层需求、服务和协议

应用层需求	传输层服务	UDP	TCP
为运行在不同主机上的进程之间提供逻辑通信	进程间交付	✓	✓
检测报文段是否出错	差错检测	✓	✓
解决丢包、差错问题	可靠传输	✗	✓
解决乱序问题	按序交付	✗	✓
解决接收缓存溢出问题	流量控制	✗	✓
应对网络拥塞	拥塞控制	✗	✓



# TCP讲解内容

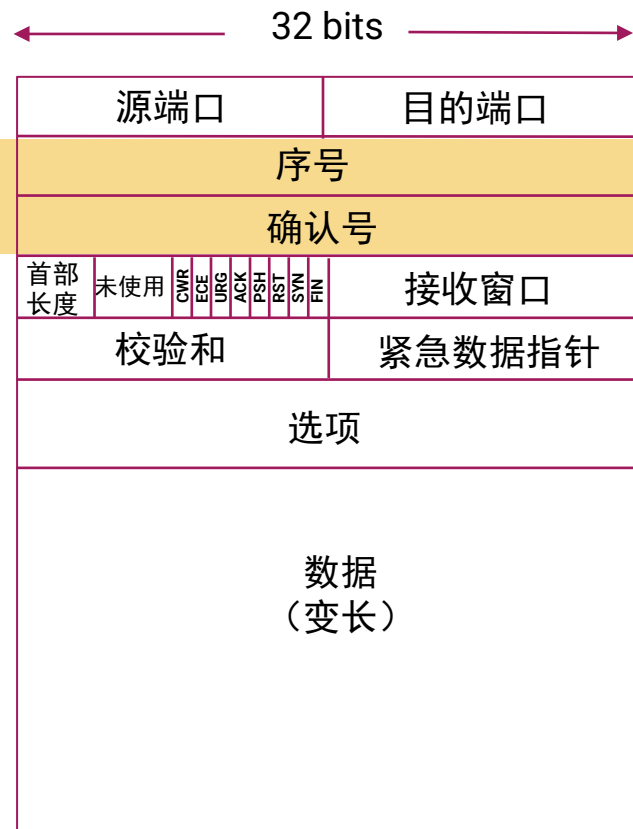
---

- 报文段结构
- 超时间隔
- 流量控制
- 连接管理

# TCP报文段结构

- 固定首部长度：20字节
- 选项：变长，长度由首部长度可知

基于字节流

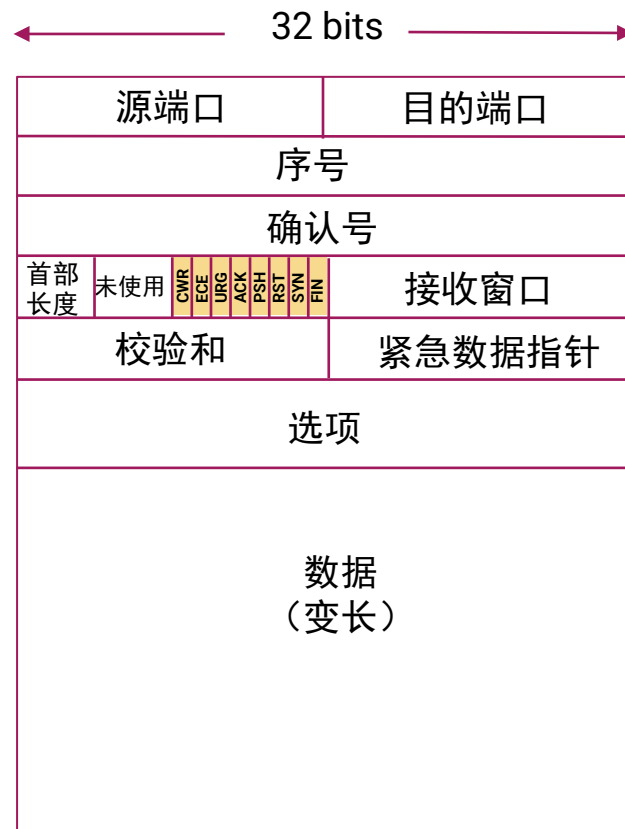


# TCP报文段结构

## ■ 选项字段

- ACK比特：指示确认字段中的值有效
- SYN比特：用于建立连接
- FIN比特：用于拆除连接

CWR  
ECE  
URG  
ACK  
PSH  
RST  
SYN  
FIN



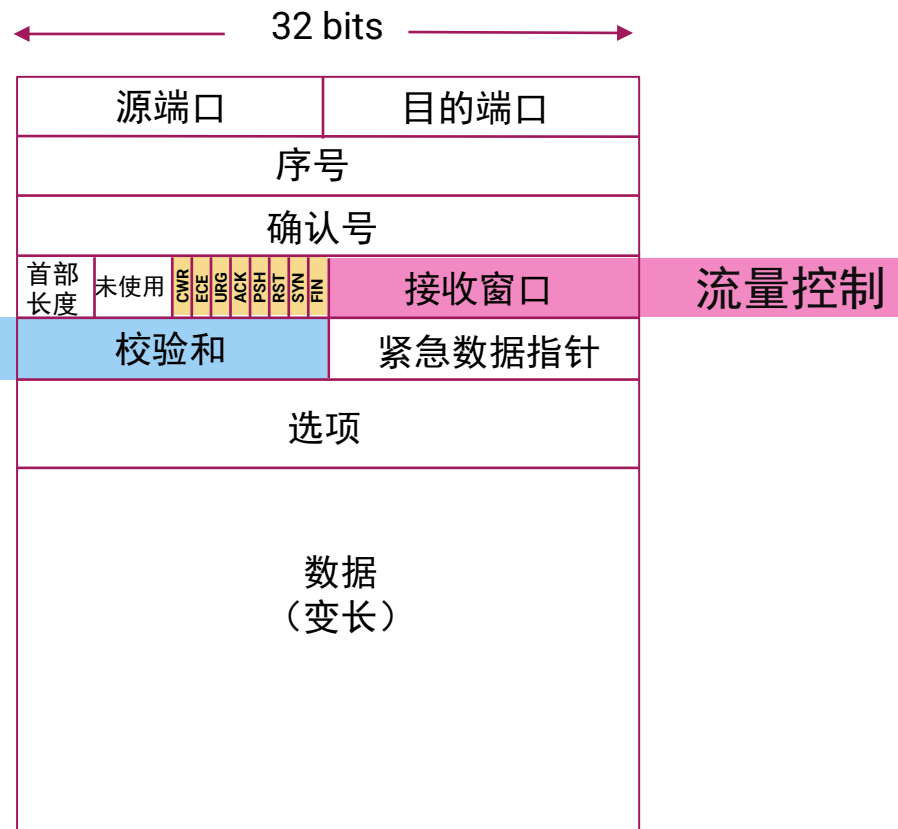
# TCP报文段结构

## ■ 选项字段

- ACK比特：指示确认字段中的值有效
- SYN比特：用于建立连接
- FIN比特：用于拆除连接

CWR  
ECE  
URG  
ACK  
PSH  
RST  
SYN  
FIN

与UDP做法相同



# TCP报文段结构

- 数据长度  $\leq$  MSS
  - Maximum Segment Size  
最大报文段长度
  - 典型值1460字节
- 链路层帧长度  $\leq$  最大传输单元MTU
  - 典型值1500字节
  - 链路层帧长度  
= 报文段数据 + TCP固定首部 + IP固定首部  
= 1460 + 20 + 20  
= 1500字节





# TCP报文段结构

- 在Windows10的powershell里执行下列指令，看看你的主机用的MTU值是多少？
- **netsh interface ipv4 show subinterfaces**



# TCP报文段结构：举例

http.pcapng

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

http

No.	Time	Source	Destination	Protocol	Length	Info
565	32.589977	192.168.2.178	184.86.198.104	HTTP	504	GET /Xplore/home.jsp HTTP/1.1

> Frame 565: 504 bytes on wire (4032 bits), 504 bytes captured (4032 bits) on interface \Device\NPF\_{...}

> Ethernet II, Src: ASUSTekC\_..., Dst: ASUSTekC\_...

> Internet Protocol Version 4, Src: 192.168.2.178, Dst: 184.86.198.104

▼ Transmission Control Protocol, Src Port: 3073, Dst Port: 80, Seq: 1, Ack: 1, Len: 450

- Source Port: 3073
- Destination Port: 80 **源端口号和目的端口号**
- [Stream index: 17]
- [TCP Segment Len: 450]
- Sequence number: 1 (relative sequence number)
- Sequence number (raw): 817913973 **序列号**
- [Next sequence number: 451 (relative sequence number)]
- Acknowledgment number: 1 (relative ack number)
- Acknowledgment number (raw): 1952848251 **确认号**
- 0101 .... = Header Length: 20 bytes (5) **报头长度**
- > Flags: 0x018 (PSH, ACK) **标志位**
- Window size value: 1026 **窗口大小**
- [Calculated window size: 262656]
- [Window size scaling factor: 256]
- Checksum: 0x43f6 [unverified] **校验和**
- [Checksum Status: Unverified]
- Urgent pointer: 0
- > [SEQ/ACK analysis]
- > [Timestamps]
- TCP payload (450 bytes) **数据**
- > Hypertext Transfer Protocol

0030 04 02 43 f6 00 00 47 45 54 20 2f 58 70 6c 6f 72 ..C. GET /Xplor

Urgent pointer (tcp.urgent\_pointer), 2 byte(s) | 分组: 8089 · 已显示: 5 (0.1%) · 已丢弃: 0 (0.0%) | 配置: Default



# 第三章知识点汇总

---

- 了解TCP报文段首部主要字段的含义



# TCP讲解内容

---

- 报文段结构
- **超时间隔**
- 流量控制
- 连接管理



# TCP超时间隔

- 如何设置TCP超时间隔？
- 须大于往返时延（RTT）
  - 如果太短，误判丢包，产生不必要的重传
  - 如果太长，等待时间就很长
- 如何预测RTT？
  - 在报文段传输时，采集样本
  - 估计RTT为近期样本的统计平均



# TCP超时间隔

- 指数加权滑动平均(EWMA)计算RTT估计

$$\text{RTT估计} = (1 - \alpha) * \text{RTT估计} + \alpha * \text{RTT样本}$$

- 参数典型取值  $\alpha = 1/8$

指数加权滑动平均: 单个样本的权值随时间递减, 指数衰减

$$E_0 = 0$$

$$E_1 = \alpha S_1$$

$$E_2 = (1 - \alpha)E + \alpha S_2 = (1 - \alpha)\alpha S_1 + \alpha S_2$$

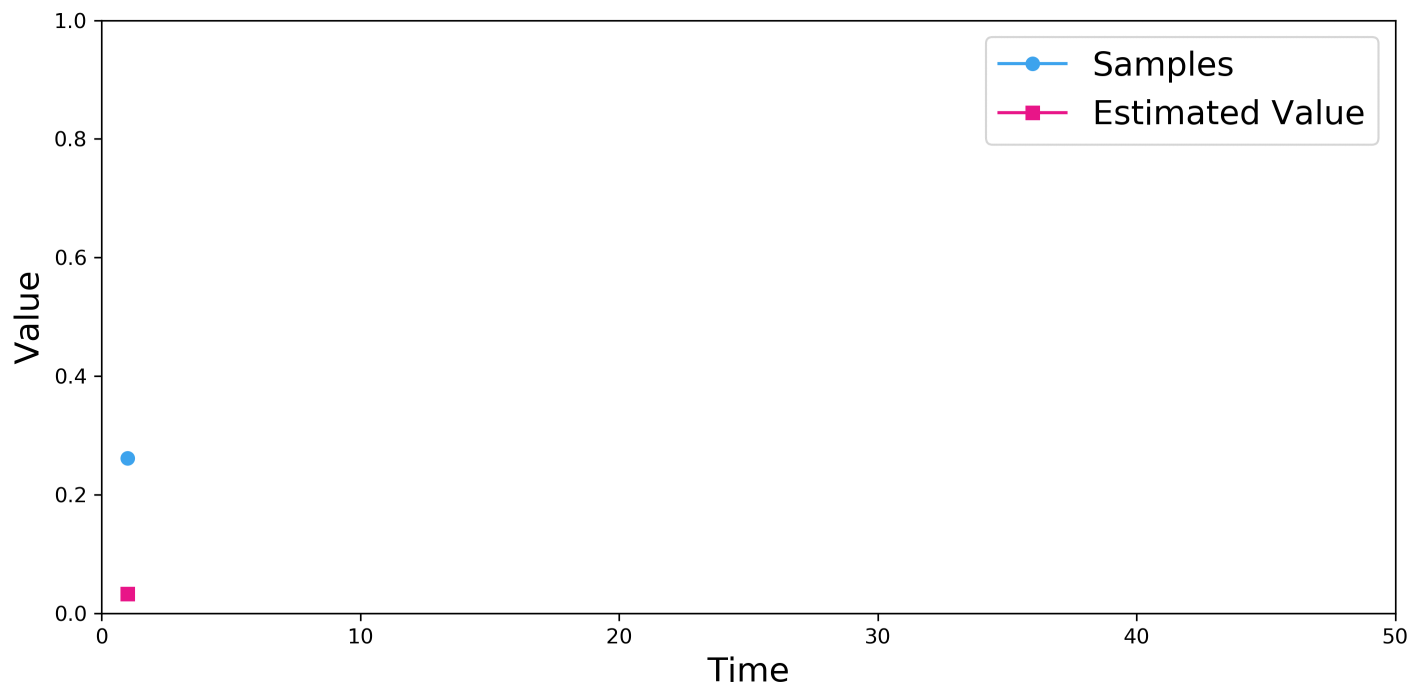
$$E_3 = (1 - \alpha)^2 \alpha S_1 + (1 - \alpha)\alpha S_2 + \alpha S_3$$

# TCP超时间隔

- 指数加权滑动平均(EWMA)计算RTT估计

$$\text{RTT估计} = (1 - \alpha) * \text{RTT估计} + \alpha * \text{RTT样本}$$

- 参数典型取值 $\alpha = 1/8$

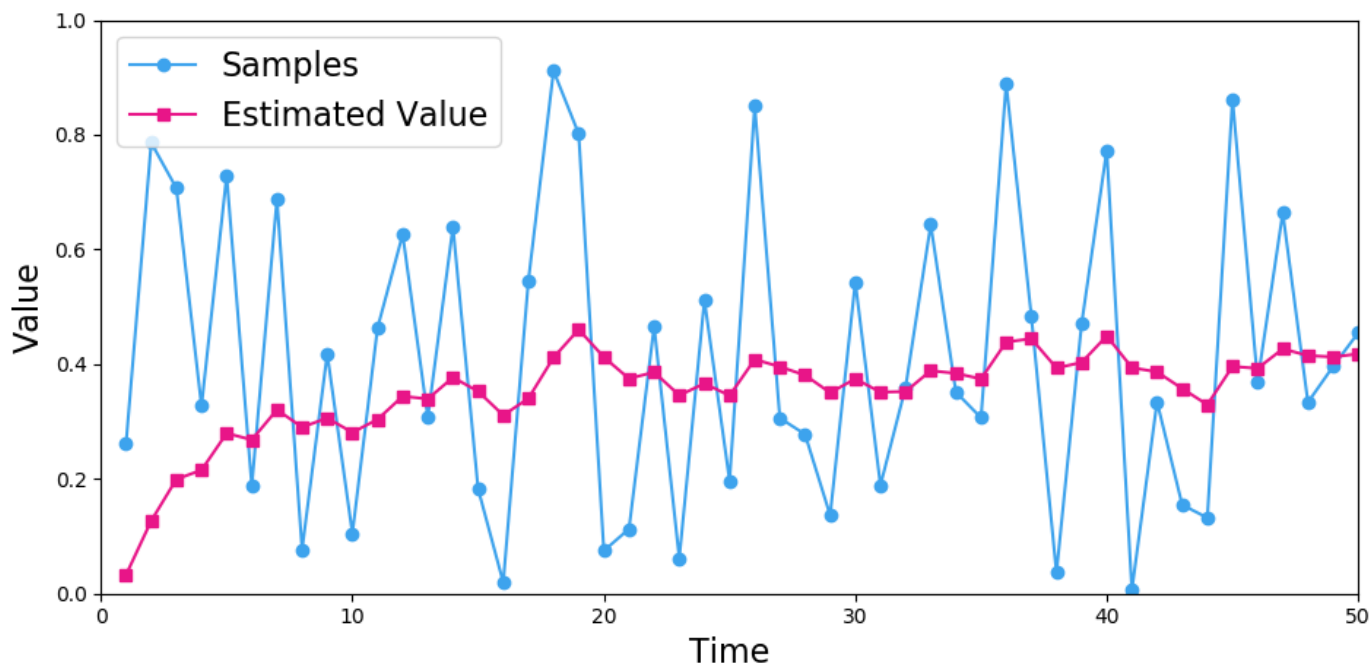


# TCP超时间隔

- 指数加权滑动平均(EWMA)计算RTT估计

$$\text{RTT估计} = (1 - \alpha) * \text{RTT估计} + \alpha * \text{RTT样本}$$

- 参数典型取值  $\alpha = 1/8$







# TCP超时间隔

- 超时间隔要大于EstimatedRTT，大多少好呢？
- RTT偏差（典型取值 $\beta=0.25$ ）

$$\text{RTT偏差} = (1 - \beta) * \text{RTT偏差} + \beta * |\text{RTT样本} - \text{RTT估计}|$$

- 超时间隔

$$\text{超时间隔} = \text{RTT估计} + 4 * \text{RTT偏差}$$

- RTT样本波动大时，余量大；
- RTT样本波动小时，余量小。



# 第三章知识点汇总

---

- 理解TCP定时器超时间隔的设置方法



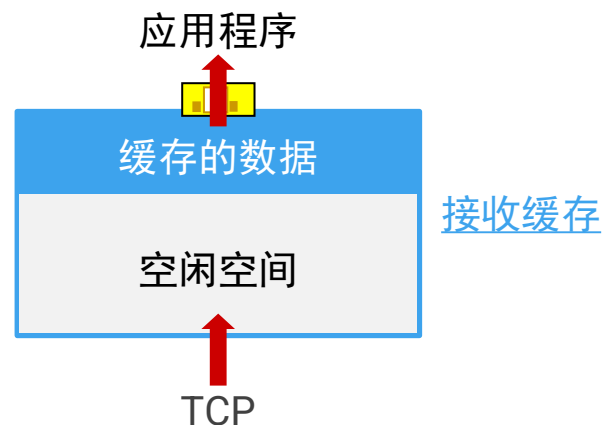
# TCP讲解内容

---

- 报文段结构
- 超时间隔
- **流量控制**
- 连接管理

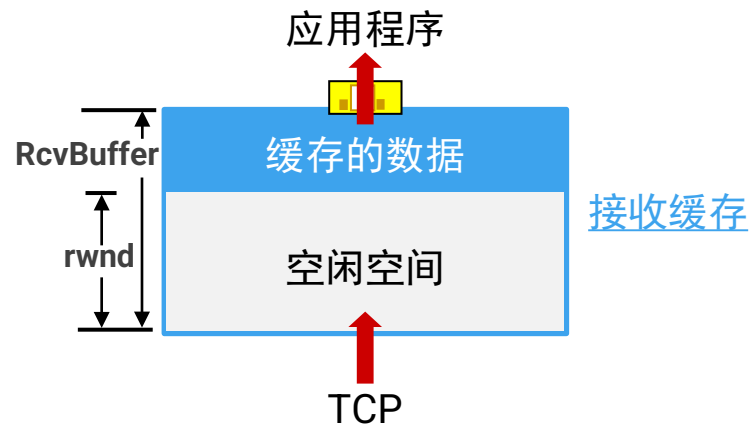
# TCP流量控制

- TCP向接收缓存**存储**数据
  - 速度主要由发送方决定
- 应用层从接收缓存**读取**数据
  - 速度由应用程序决定
- 如果应用层读取太慢，会发生什么呢？
  - 接收缓存很快就会被塞满
- 流量控制
  - 接收方控制发送方的发送速度，避免接收方的接收缓存溢出。



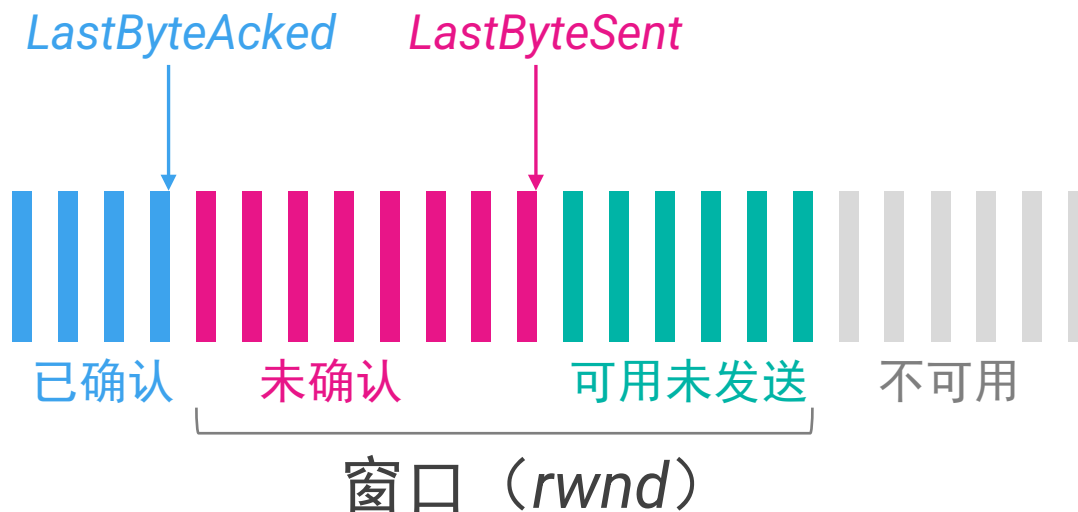
# TCP流量控制

- 接收方计算接收窗口（rwnd），将其放在TCP首部随数据发给发送方
- 发送方限制**已发送未确认的字节数**（窗口）不超过rwnd
- 接收缓存（RcvBuffer）的容量由socket的选项决定



# TCP流量控制

- 已发送未确认的字节数  $\leq rwnd$
- $LastByteSent - LastByteAcked \leq rwnd$



图示：发送方的序号空间



# 第三章知识点汇总

---

- 理解流量控制技术的目的
- 理解流量控制技术的原理



# TCP讲解内容

---

- 报文段结构
- 超时间隔
- 流量控制
- 连接管理

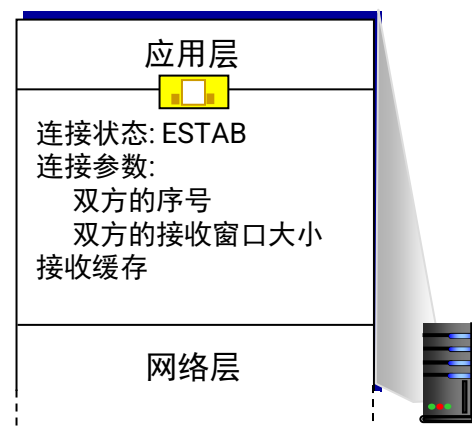


# TCP连接管理

- 在交换数据之前，发送方和接收方要先“握手”
  - 双方同意建立连接
  - 分配缓存和初始化变量



`clientSocket.connect((serverName,  
serverPort))`

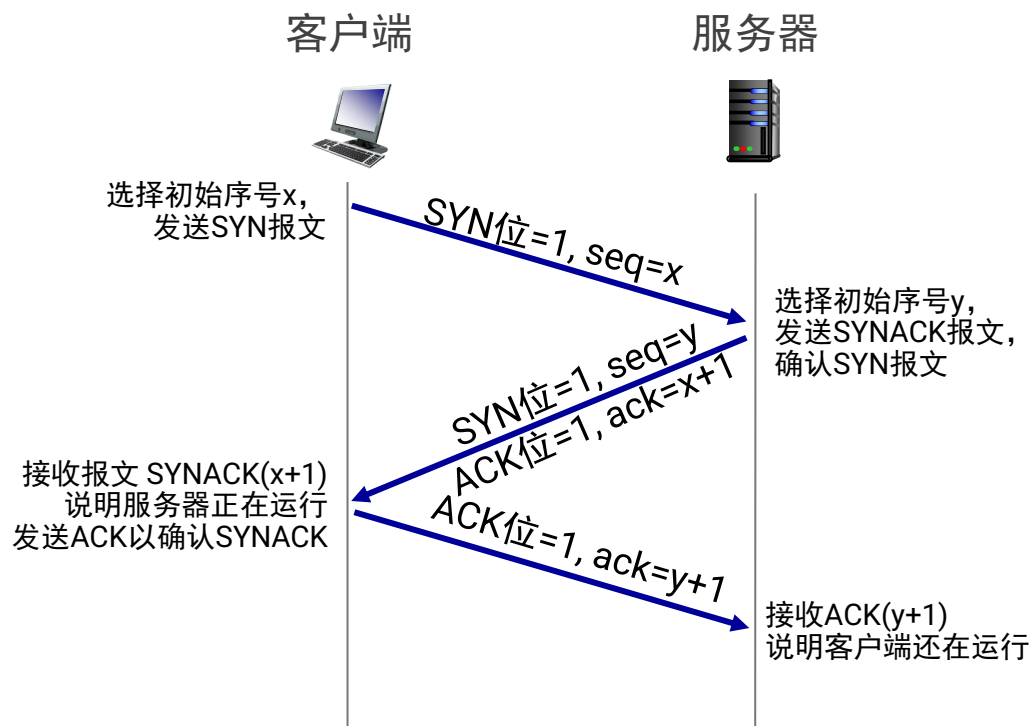


`connectionSocket, addr =  
serverSocket.accept()`

# TCP建立连接

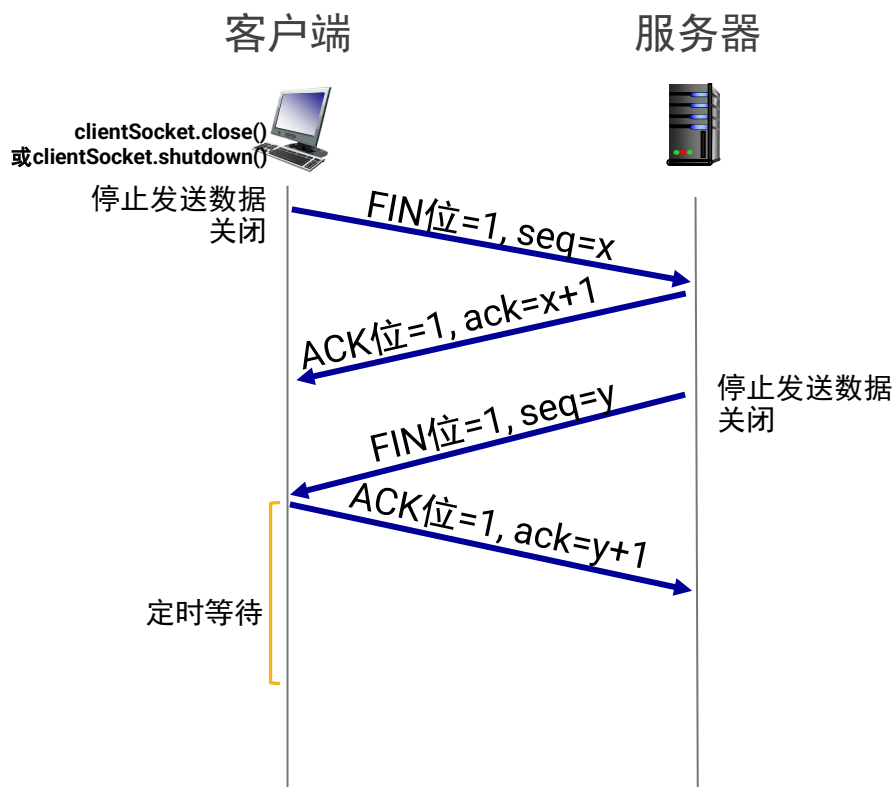
## ■ 三次握手

- SYN报文虽然不带数据，但在计算ack时仍然算一个字节



# TCP关闭连接

- 双方分别关闭连接
  - ACK分组和FIN分组可以合并





# 第三章知识点汇总

---

- 理解TCP建立连接的步骤
- 理解TCP关闭连接的步骤

*That I exist is a perpetual surprise which is life.*

我的存在，是一个永恒的神奇，这就是生活。

——*Tagore*