一图看懂|图解TCP/UDP

请叫我大师兄 极客重生 2022-11-03 06:04 Posted on 广东

收录于合集

#深入理解网络

41个

TCP - Summary Cheat sheet

Version 2

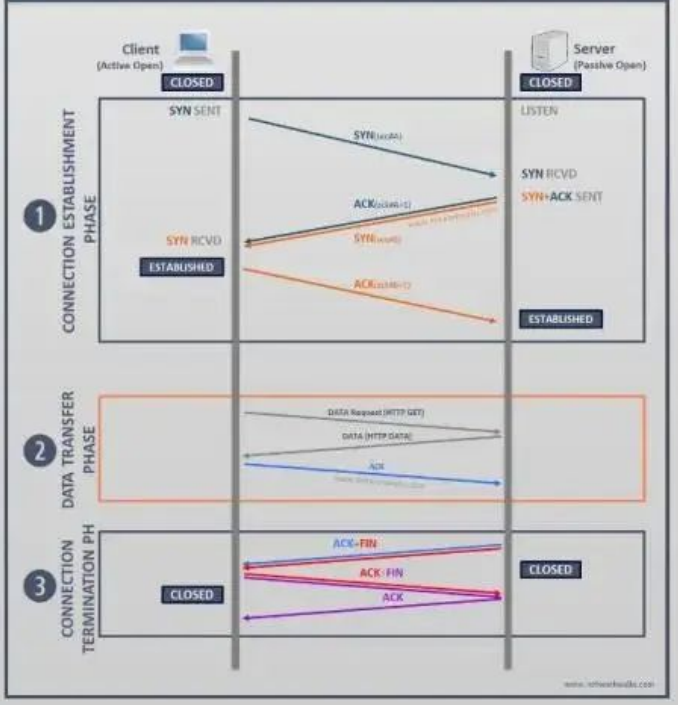
network Walks

Phase1 3-way handshake is completed & trust relationship is built b/w Sender/Rec

Phase2 The connection is opened and the participant devices start sending data using the agreed sequence and acknowledge numbers that have been agreed in phase1

Phase3 Connection is terminated with FIN flags once all Data transfer is completed

TCP 3-way Handshake Process



TCP Message Types

Message	Description
SYN (Synchronize message)	Used to initiate and establish a connection. It is used to synchronize sequence numbers between devices. **SYN bit =1** in the TCP Header
ACK (Acknowledgement message)	Used to confirm to the other side that it has received the SYN **ACK bit =1** in the TCP Header
SYN-ACK (Synchronize & ACK message)	SYN message from local device & ACK of the previous packet. **SYN bit =1,ACK bit=1** in the TCP Header
FIN (Finish)	Used to terminate a connection. **FIN bit =1** in the TCP Header

TCP CALLS

Active OPEN A device using TCP takes the active role and initiates the connection by sending a TCP SYN message to start the connection. The Device in Active OPEN state is called Client

Passive OPEN Device is waiting for an active OPEN from other. It does not generate any TCP message segment. The Device in Passive OPEN state is called Server

Transport Layer Ports

Category	Range	Comments
Well Known Ports	0 – 1023	Used by system processes e.g. FTP(21)
Registered Ports	1024 - 49151	For specific services e.g. Port 8080
Private Ports	49152 – 65535	For Private purposes

Important TCP/UDP Ports

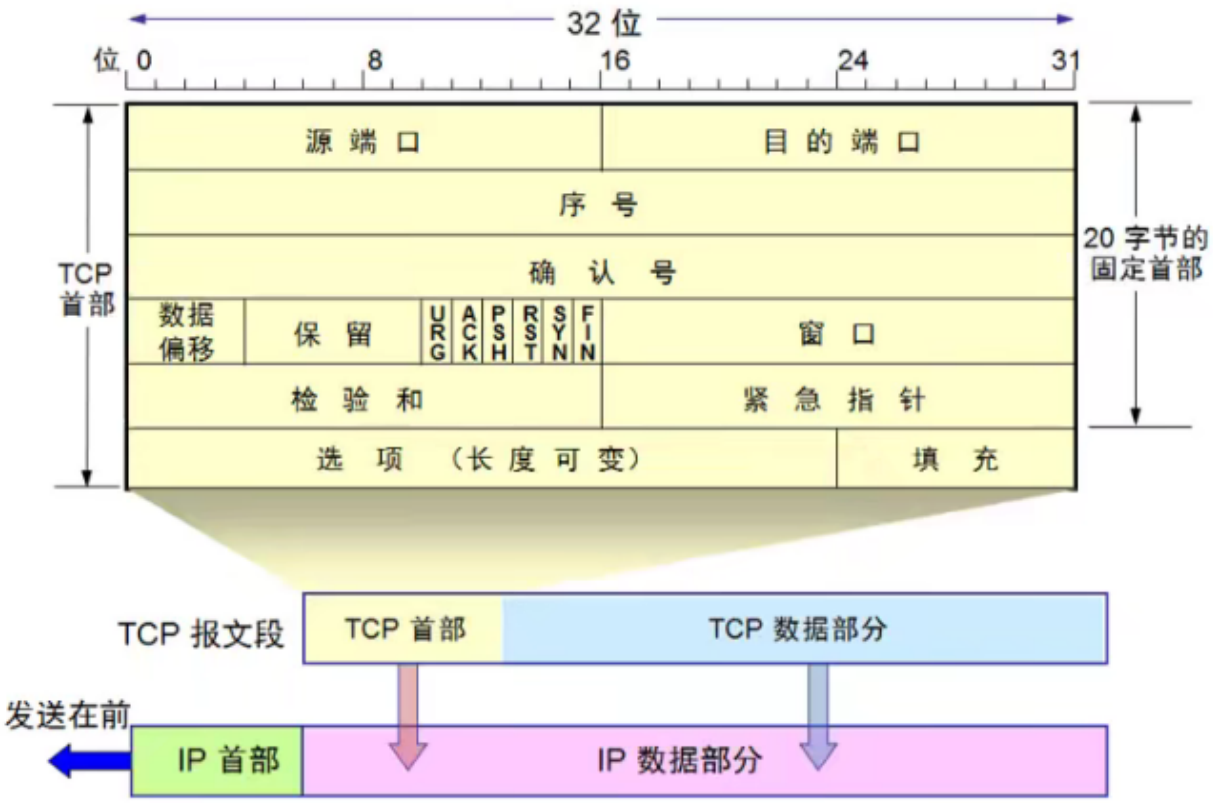
Port Number	Protocol	Application
20	TCP	FTP data
21	TCP	FTP control
23	TCP	Telnet
25	TCP	SMTP
53	TCP	DNS
67, 68	UDP	DHCP
69	UDP	TFTP
80	TCP	HTTP/WWW
110	TCP	POP3
143	TCP	IMAP
443	TCP	SSL
16,384-32,767	UDP	RTP-based Voice/VideoRTP, Cisco Voice

TCP States

State	Description
CLOSED	In-active or Initial state where not TCP activity has begun yet
LISTEN	The device is waiting for contact request
SYN-SENT	The device waits to receive an ACK to the SYN it has sent to the other side
SYN+ACK SENT	The device sends an ACK that it has received the SYN. Also, it sends its own SYN request & waits to receive an ACK from the other side
SYN RCVD	The device has received the SYN for the ACK it sent previously
ESTABLISHED	TCP Handshake has been completed/Established & the device is ready for data transfer now

一、TCP

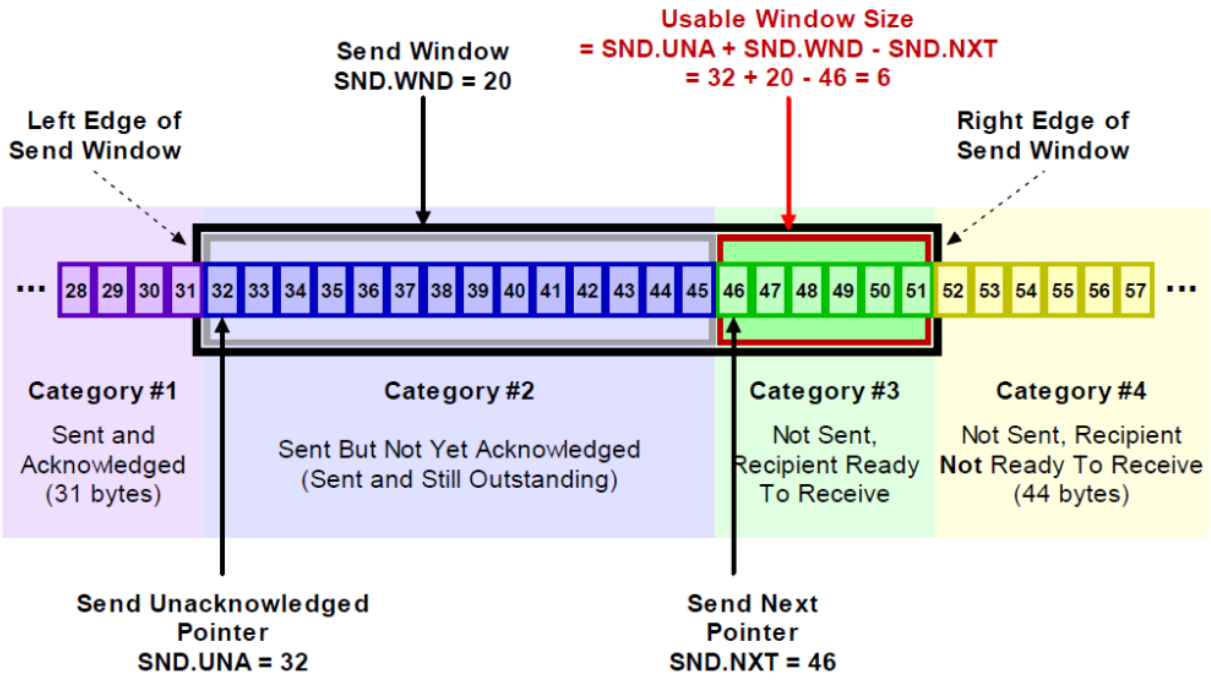
1、TCP首部



- **源端口**(Source Port): 使用 TCP 协议传输数据的端口号
- **目的端口**(Destination Port): 数据传输目的主机所对应的端口号
- **序号**(Sequence Number): 表示在该报文段中的数据相对于要发送的所有数据中的偏移量(从上层传下来的数据一般在传输层就已经切分了, 这样如果有一个报文段丢失/出错的话, 发送端就只需要传输对应的报文段即可(TCP 差错控制). 若是到网络层或数据链路层再切分, 则若出错, 需要重新发送整个数据)
- **确认号**(Acknowledgement Number): 表示期望对方发送过来的报文段的序号(Sequence Number)
- **数据偏移**(Data Offset): 指数据部分在整个 TCP 报文段中的偏移量(即 TCP 首部的长度)
- **控制位**
  - **URG**(Urgent): 表示是否有紧急情况, 当值为 1 时, 从"数据偏移"开始的"紧急指针"个字节会优先发送
  - **ACK**(Acknowledgement): 当值为 1 时, 对应确认号(ack)表示期望对方接下来发送过来的数据的序号(seq)
  - **PSH**(Push): 当 PSH 值为 1 时, 接收到该报文段的主机会尽快将数据交付给应用程序, 而不会等到缓存满了之后再交付给上层
  - **RST**(Reset flag): 当值为 1 时, 表示网络状态不好, 需要释放连接, 并重新建立连接

    - **SYN**(Synchronize flag): 当值为 1 时, 表示期望与对方建立连接
    - **FIN**(Final flag): 释放连接
- **16位窗口大小**(Window Size): 接收/发送窗口的大小，流量控制使用，如果窗口大小为0，可以发送窗口探测
- **16位校验和**(Checksun): 校验和用来做差错控制，TCP校验和的计算包括TCP首部、数据和其它填充字节。在发送TCP数据段时，由发送端计算校验和，当到达目的地时又进行一次检验和计算。如果两次校验和一致，说明数据是正确的，否则将认为数据被破坏，接收端将丢弃该数据
- **16位紧急指针**：仅在URG控制位为 1 时有效。表示紧急数据的末尾在 TCP 数据部分中的位置。通常在暂时中断通信时使用（比如输入 Ctrl + C）

2、流量控制



流量控制，就是让发送方的发送速率不要太快，要让接收方来得及接收

利用滑动窗口机制可以很方便地在tcp连接上实现对发送方的流量控制

TCP接收方利用自己的接收窗口的大小来限制发送方发送窗口的大小

TCP接受方的窗口可以划分成四个部分：

1、已经接收并且已经确认的TCP段；

2、已经接收但是没有确认的TCP段；

3、还未接收但是发送方已经发送的TCP段；

4、还未接收但是发送也不允许发送的TCP段。

重传计时器

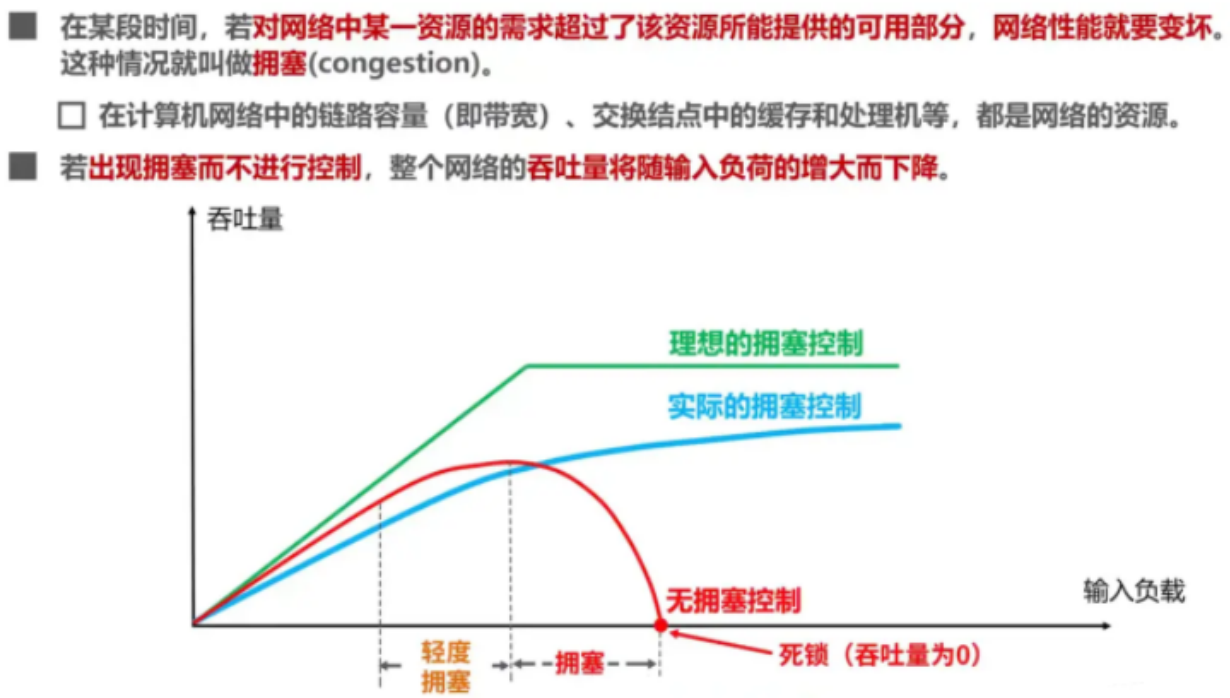
TCP发送方收到接收方的零窗口通知后，应启动持续计时器。持续计时器超时后，向接收方发送零窗口探测报文

即使接收窗口为0，接收方也会接收：零窗口探测报文段、确认报文段、携带紧急数据的报文段

TCP发送方的发送窗口大小 = Math.min(自身拥塞窗口大小, TCP接收方的接收窗口大小)

3、拥塞控制

什么是拥塞



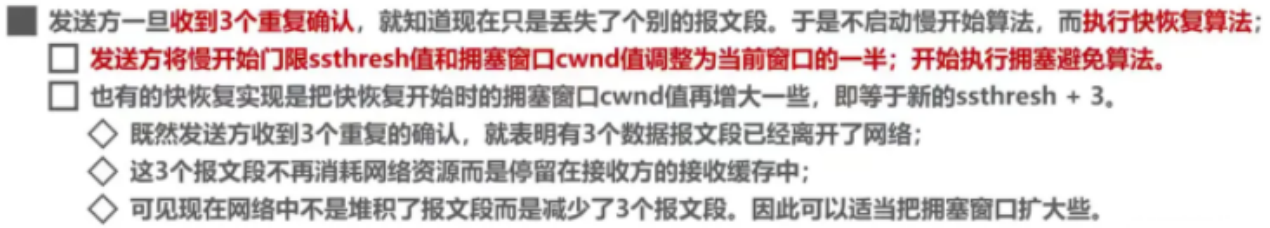
假定条件

数据是单方向发送，而另一方向只传送确认 接收方总是有足够大的缓存空间，因而发送方发送窗口的大小由网络的拥塞程度来决定 以最大报文段MSS的个数为讨论问题的单位，而不是以字节为单位

慢开始 + 拥塞避免算法

MSS：TCP最大报文段 ssthresh：慢开始门限 cwnd：拥塞窗口 swnd：发送窗口 rtt：每次往返时间

快重传



慢开始 + 拥塞避免算法中，发送方把拥塞窗口cwnd又设置为1，并错误地启动慢开始算法，降低了传输效率

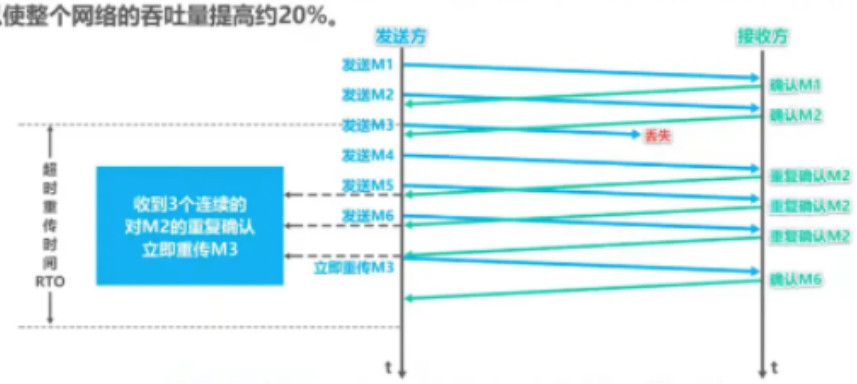
- 采用快重传算法可以**让发送方尽早知道发生了个别报文段的丢失。**
- 所谓快重传，就是使发送方**尽快进行重传**，而**不是等超时重传计时器超时**再重传。
  - 要求接收方不要等待自己发送数据时才进行捎带确认，而是要**立即发送确认**；
  - 即使收到了失序的报文段也要立即发出对已收到的报文段的**重复确认**。
  - 发送方一旦**收到3个连续的重复确认**，就将相应的报文段**立即重传**，而不是等该报文段的超时重传计时器超时再重传。

收到3个重复确认

接收方收到失序的报文段，立即发出重复确认

发送方收到3个连续的重复确认，立即重传

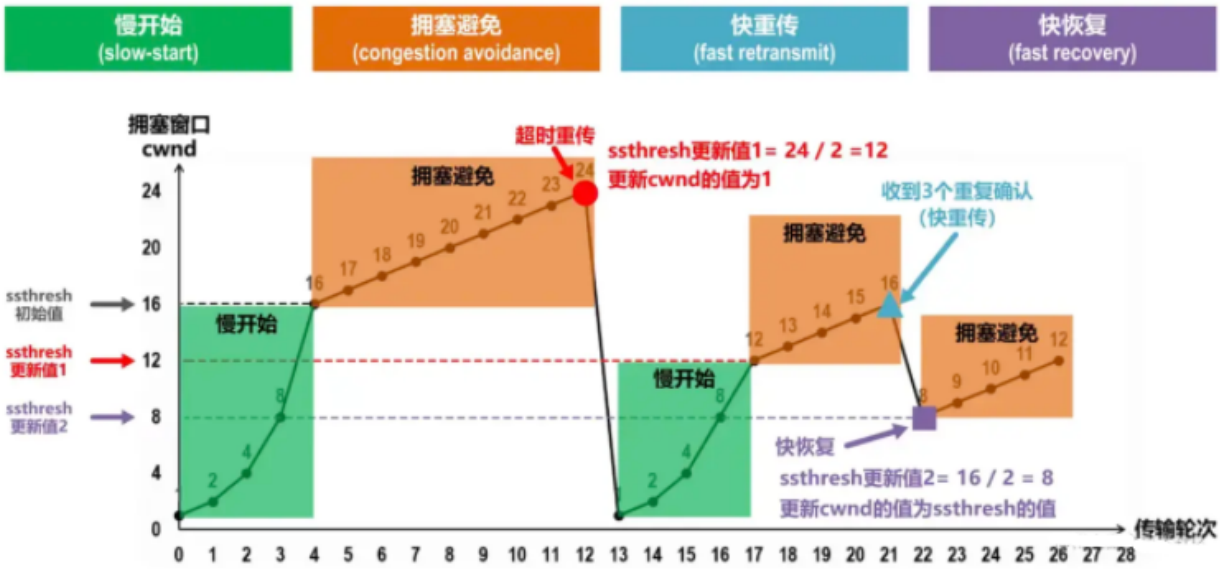
- 对于个别丢失的报文段，发送方不会出现超时重传，也就不会误认为出现了拥塞（进而降低拥塞窗口cwnd为1）。使用快重传可以使整个网络的吞吐量提高约20%。



快恢复

- 发送方一旦**收到3个重复确认**，就知道现在只是丢失了个别的报文段。于是不启动慢开始算法，而**执行快恢复算法**；
  - **发送方将慢开始门限ssthresh值和拥塞窗口cwnd值调整为当前窗口的一半；开始执行拥塞避免算法。**
  - 也有的快恢复实现是把快恢复开始时的拥塞窗口cwnd值再增大一些，即等于新的ssthresh + 3。
    - 既然发送方收到3个重复的确认，就表明有3个数据报文段已经离开了网络；
    - 这3个报文段不再消耗网络资源而是停留在接收方的接收缓存中；
    - 可见现在网络中不是堆积了报文段而是减少了3个报文段。因此可以适当把拥塞窗口扩大些。

慢开始 + 拥塞避免+快重传 + 快恢复结合


慢开始
(slow-start)
拥塞避免
(congestion avoidance)
快重传
(fast retransmit)
快恢复
(fast recovery)
拥塞窗口
cwnd
超时重传
ssthresh更新值1= 24 / 2 =12
更新cwnd的值为1
收到3个重复确认
(快重传)
拥塞避免
拥塞避免
拥塞避免
慢开始
慢开始
ssthresh
初始值
ssthresh
更新值1
ssthresh
更新值2
快恢复
ssthresh更新值2= 16 / 2 = 8
更新cwnd的值为ssthresh的值
传输轮次

4、TCP状态机

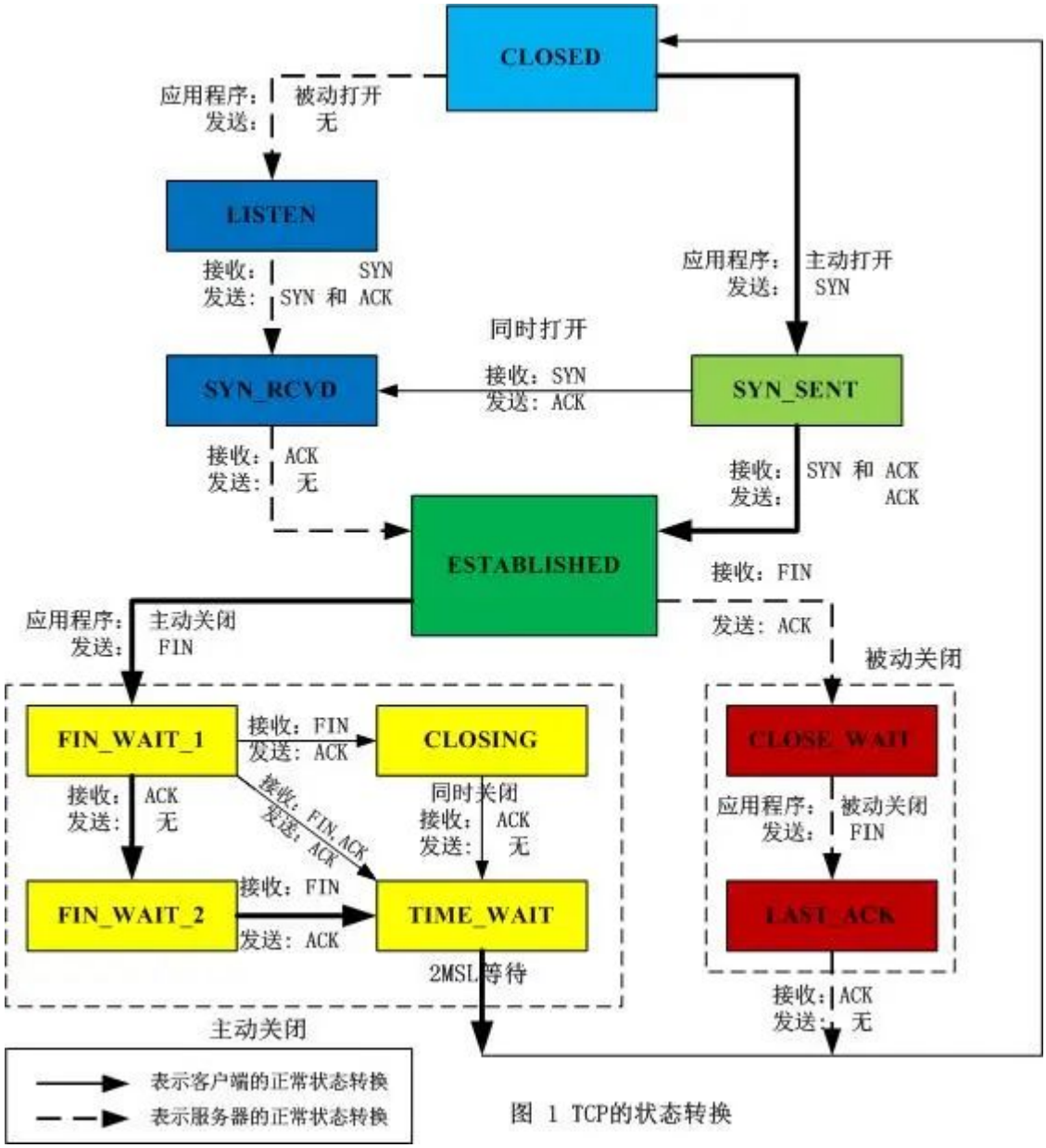


图 1 TCP的状态转换

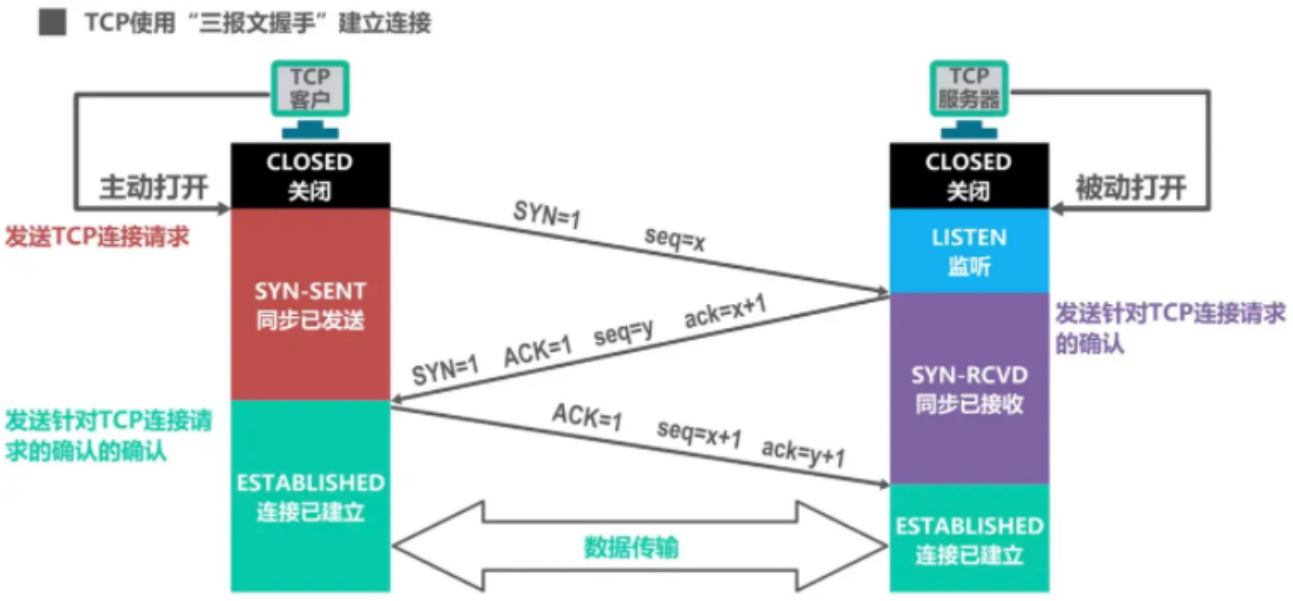
- **CLOSED**：表示初始状态。对服务端和客户端双方都一样。
- **LISTEN**：表示监听状态。服务端调用了 listen 函数使其处于监听状态，此时可以开始 accept （接收）客户端的连接。
- **SYN_SENT**：表示客户端已经发送了 SYN 报文段，则会处于该状态。当客户端调用 connect 函数发起连接请求时，首先发 SYN 给服务端，然后自己进入 SYN_SENT 状态，并等待服务端发送 ACK+SYN 作为请求应答。
- **SYN_RCVD**：表示服务端收到客户端发送 SYN 报文段。服务端收到这个报文段后，进入 SYN_RCVD 状态，然后发送 ACK+SYN 给客户端。
- **ESTABLISHED**：表示 TCP 连接已经成功建立，通信双方可以开始传输数据。服务端发送完 ACK+SYN 并收到来自客户端的 ACK 后进入该状态，客户端收到来自服务器的 SYN+ACK 并发送 ACK 后也进入该状态。
- **FIN_WAIT_1**：表示主动关闭连接。无论哪方调用 close 函数发送 FIN 报文都会进入这个这个状态。

- **FIN_WAIT_2**：表示被动关闭方同意关闭连接。主动关闭连接方收到被动关闭方返回的 ACK 后，会进入该状态。
- **TIME_WAIT**：表示收到对方的 FIN 报文并发送了 ACK 报文，就等 2MSL 后即可回到 CLOSED 状态了。如果 FIN_WAIT_1 状态下，收到对方同时带 FIN 标志和 ACK 标志的报文时，可以直接进入 TIME_WAIT 状态，而无须经过 FIN_WAIT_2 状态。
- **CLOSING**：表示双方同时关闭连接。如果双方几乎同时调用 close 函数，那么会出现双方同时发送 FIN 报文的情况，就会出现 CLOSING 状态，表示双方都在关闭连接。
- **CLOSE_WAIT**：表示被动关闭方等待关闭。当收到对方调用 close 函数发送的 FIN 报文时，回应对方 ACK 报文，此时进入 CLOSE_WAIT 状态。
- **LAST_ACK**：表示被动关闭方发送 FIN 报文后，等待对方的 ACK 报文状态，当收到 ACK 后进入CLOSED状态

4.1 三次握手

发送端：SYN=1、seq=x
接收端：ACK=1、ack=x+1、SYN=1、seq=y
发送端：ACK=1、ack=y+1、seq=x+1

TCP规定：SYN被设置为1的报文段不能携带数据，但要消耗掉一个序号
TCP规定：普通的确认报文段如果不携带数据，则不消耗序号



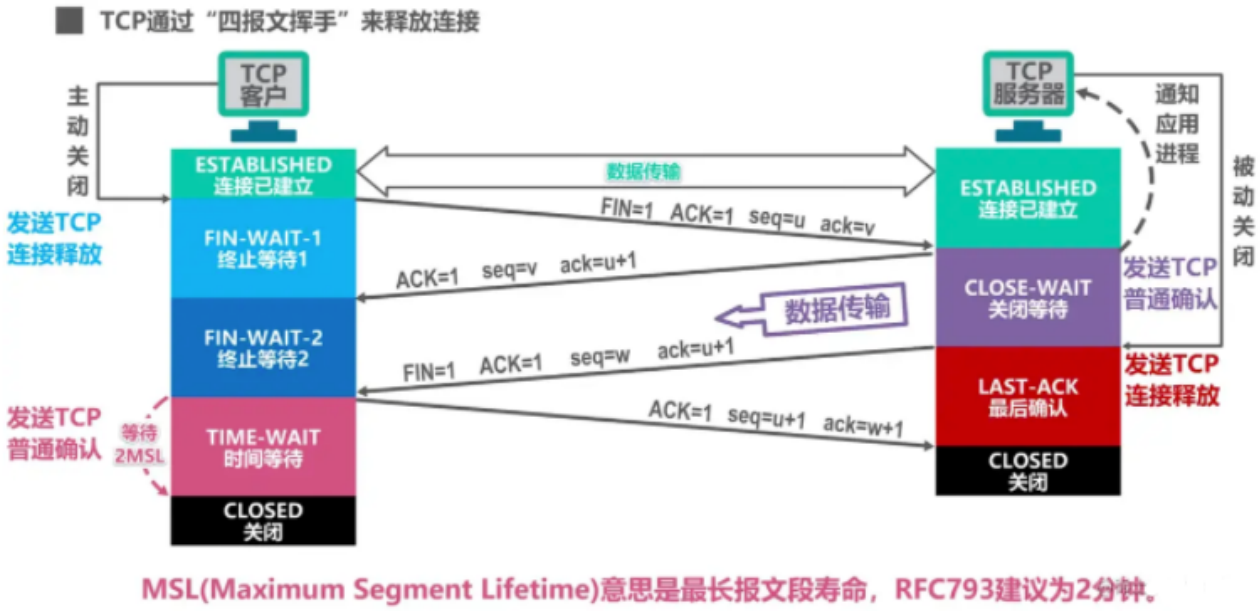
4.2 四次挥手

发送端：FIN=1，ACK=1，seq=u，ack=v（u等于发送端已传送过的数据的最后一个字节序号+1，v等于发送端之前已收到的数据的最后一字节序号+1）
接收端：ACK=1，ack=u+1，seq=v
接收端：FIN=1，ACK=1，ack=u+1，seq=w（w：半关闭情况下，可能收到了数据）
发送端：ACK=1，ack=w+1，seq=u+1

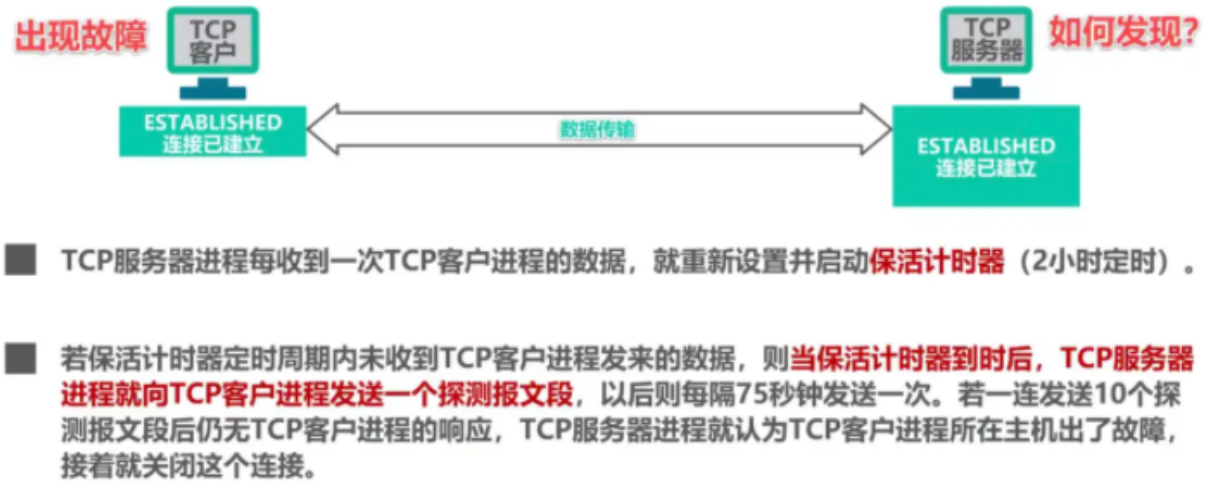
TCP规定：终止位FIN等于1的报文段，即使不携带数据，也要一个消耗掉一个序号
MSL：最长报文段寿命，建议为2分钟

为什么要等待2MSL?

如果接收端发送FIN连接释放，发送端接收后发送ACK，如果丢失，会导致接收端超时重传，而无法进入CLOSED状态



4.3 保活计时器



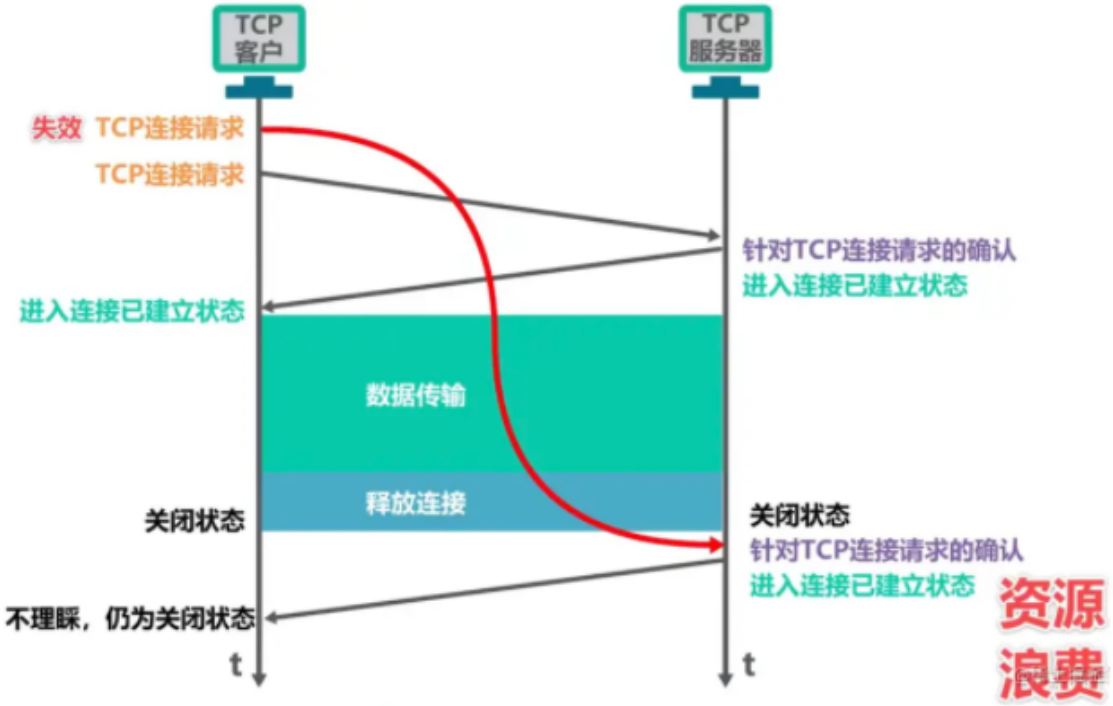
4.4 半连接队列

服务器第一次收到客户端的 SYN 之后，就会处于 SYN_RCVD 状态，此时双方还没有完全建立其连接，服务器会把此种状态下请求连接放在一个队列里，我们把这种队列称之为半连接队列。

4.5 三次握手能不能改成两次握手?

不能

TCP发送连接请求，但长时间没到达，然后触发了超时重传,又发送了一次，后建立连接，数据传输，并断开了连接,但此时之前没达到的请求报文段突然又到了接收端服务器，接收端服务器变成了ESTABLISHED状态,接收端一直在等发送端发送数据，白白浪费了主机很多资源，导致了错误



4.6 四次挥手能不能改成三次挥手?

不能

接收端可能还有数据没有发送,需要等待一段时间，发送完数据，才会发送FIN.

4.7 SYN攻击

服务器端的资源分配是在二次握手时分配的，而客户端的资源是在完成三次握手时分配的，所以服务器容易受到SYN洪泛攻击。SYN攻击就是Client在短时间内伪造大量不存在的IP地址，并向Server不断地发送SYN包，Server则回复确认包，并等待Client确认，由于源地址不存在，因此Server需要不断重发直至超时，这些伪造的SYN包将长时间占用未连接队列，导致正常的SYN请求因为队列满而被丢弃，从而引起网络拥塞甚至系统瘫痪。SYN 攻击是一种典型的 DoS/DDoS 攻击。

5、tcp 怎样保证数据正确性?

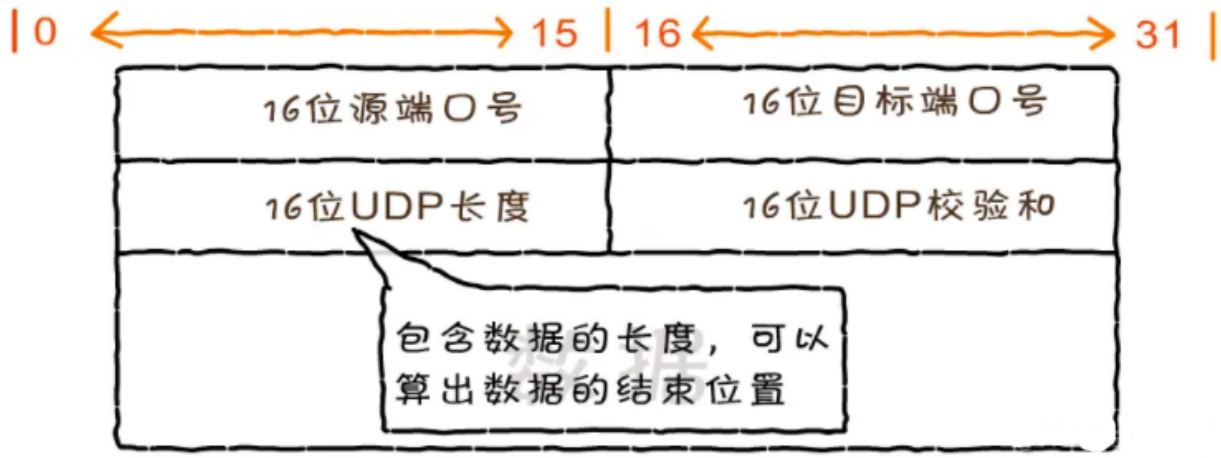
差错控制 发送的数据包的二进制相加然后取反，检测数据在传输过程中的任何变化，如果收到段的检验和有差错，TCP 将丢弃这个报文段和不确认收到此报文段。编号 + 排序 TCP 给发送的每一个包进行编号，接收方对数据包进行排序，把有序数据传送给应用层 确认 + 超时重传的机制 当 TCP 发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段。流量控制

TCP 连接的每一方都有固定大小的缓冲空间，TCP 的接收端只允许发送端发送接收端缓存区能接纳的数据。当接收方来不及处理发送方的数据，能提示发送方降低发送的速率，防止包丢失。TCP 使用的流量控制协议是可变大小的滑动窗口协议。

拥塞控制

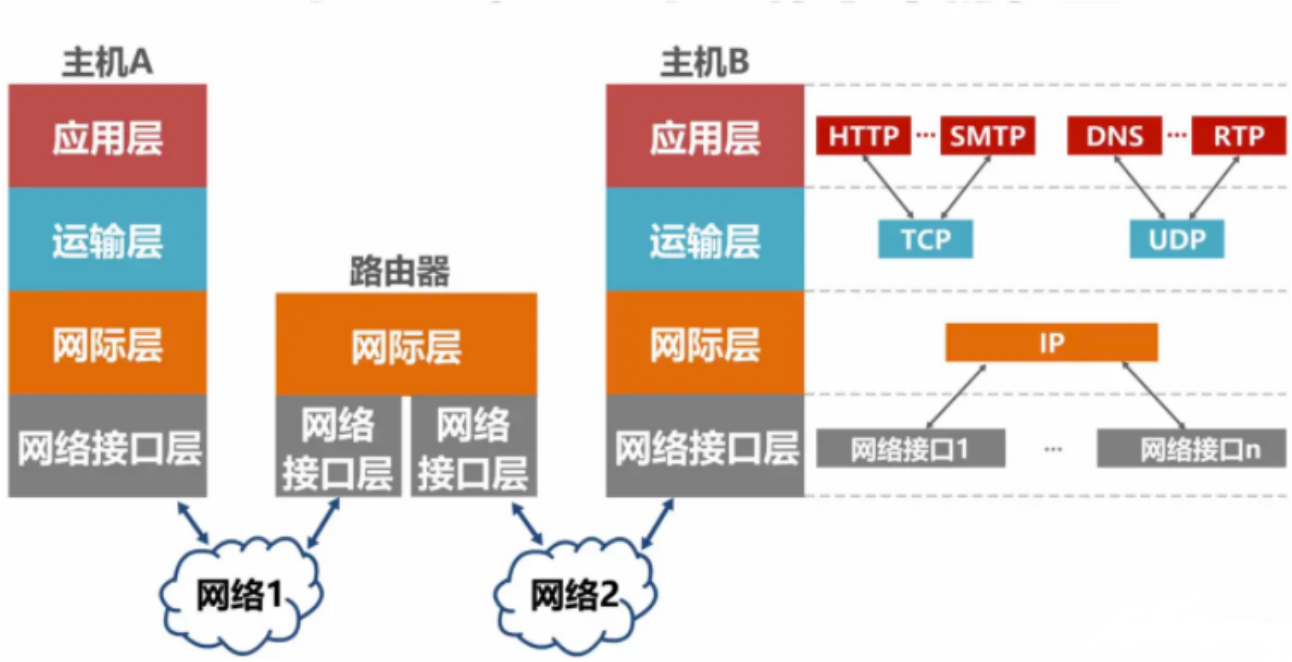
当网络拥塞时，减少数据的发送。发送方有拥塞窗口，发送数据前比对接收方发过来的接收窗口，取两者的最小值---慢启动、拥塞避免、拥塞发送、快速恢复

二、UDP

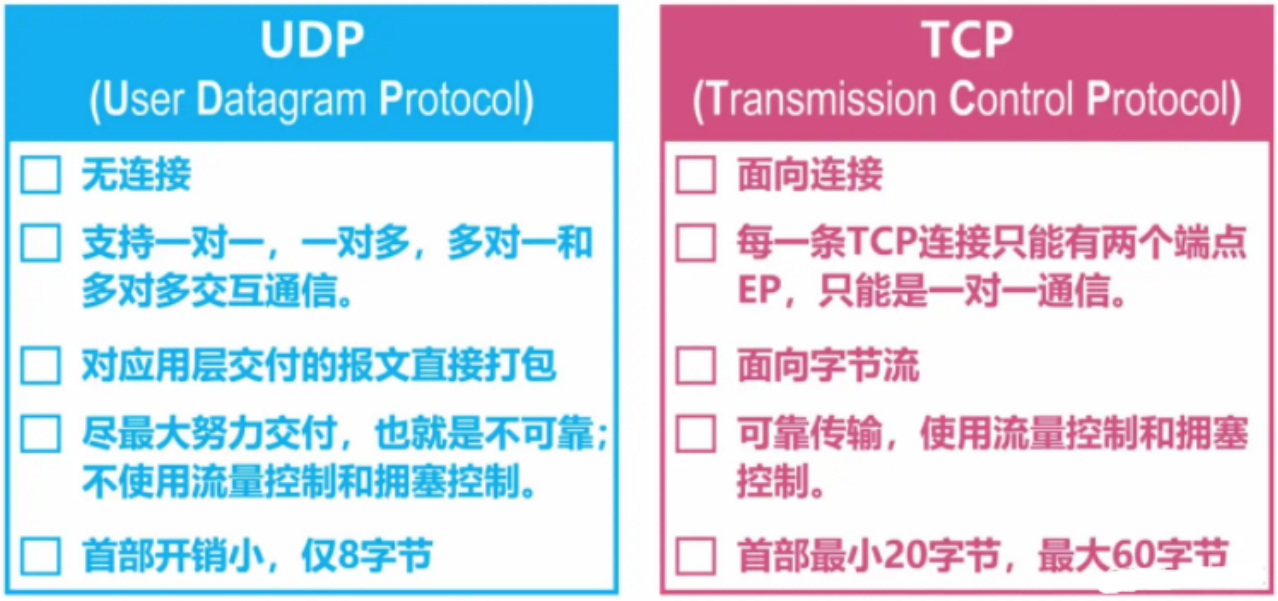


三、TCP/UDP对比

TCP/IP协议架构

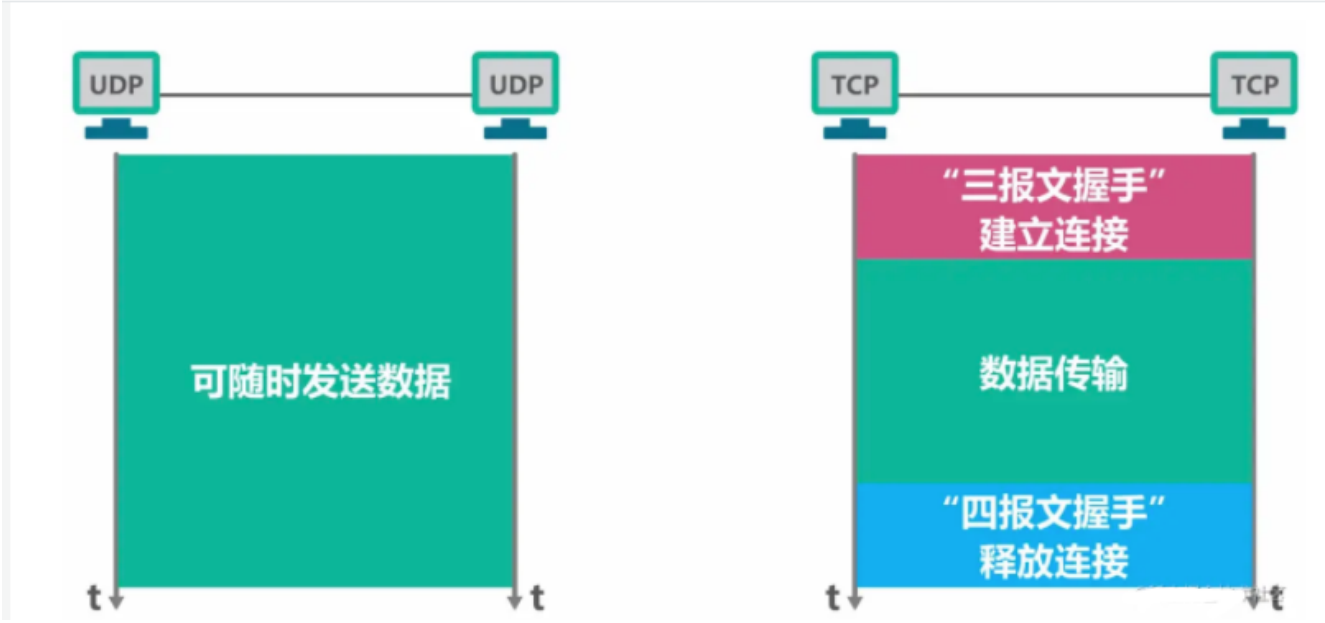


对比



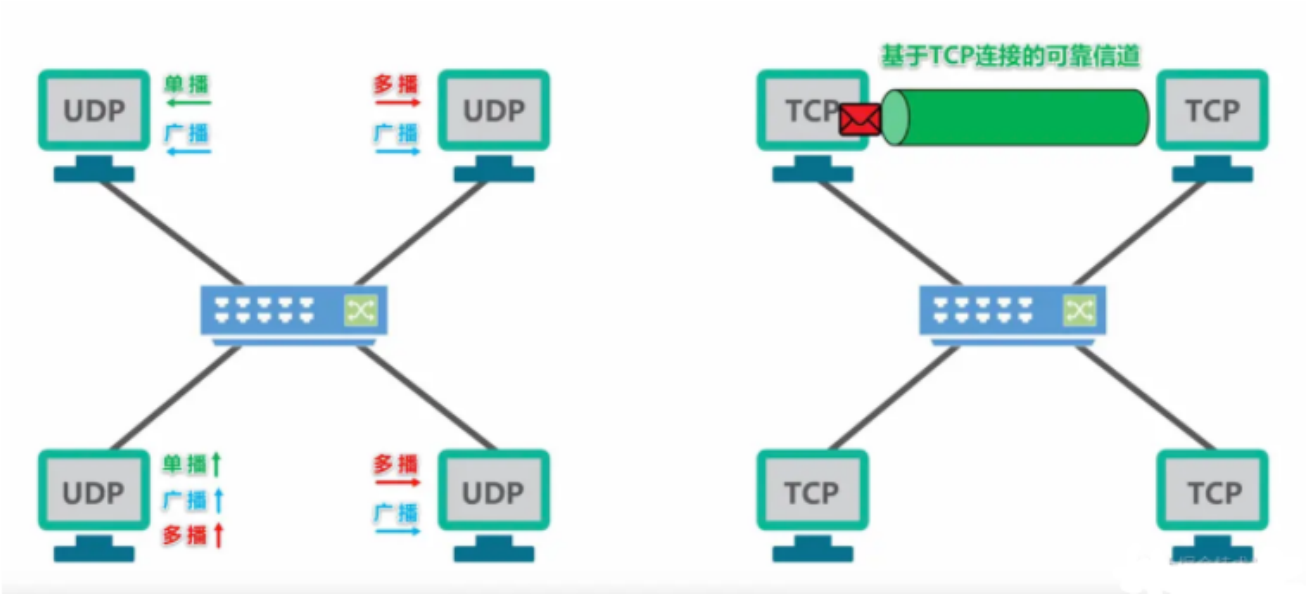
1、是否面向连接

UDP：无连接
TCP：面向连接（三次握手，四次挥手）



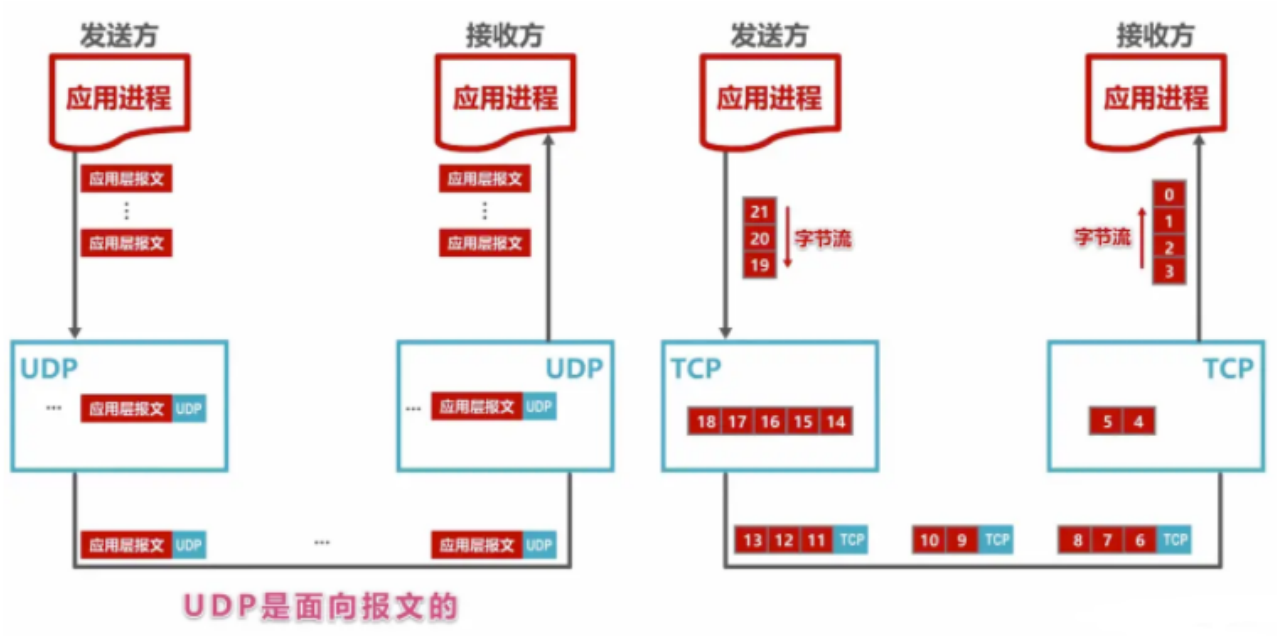
2、是否支持广播和多播

UDP：支持一对一，一对多，多对一和多对多交互通信
TCP：只能一对一通信



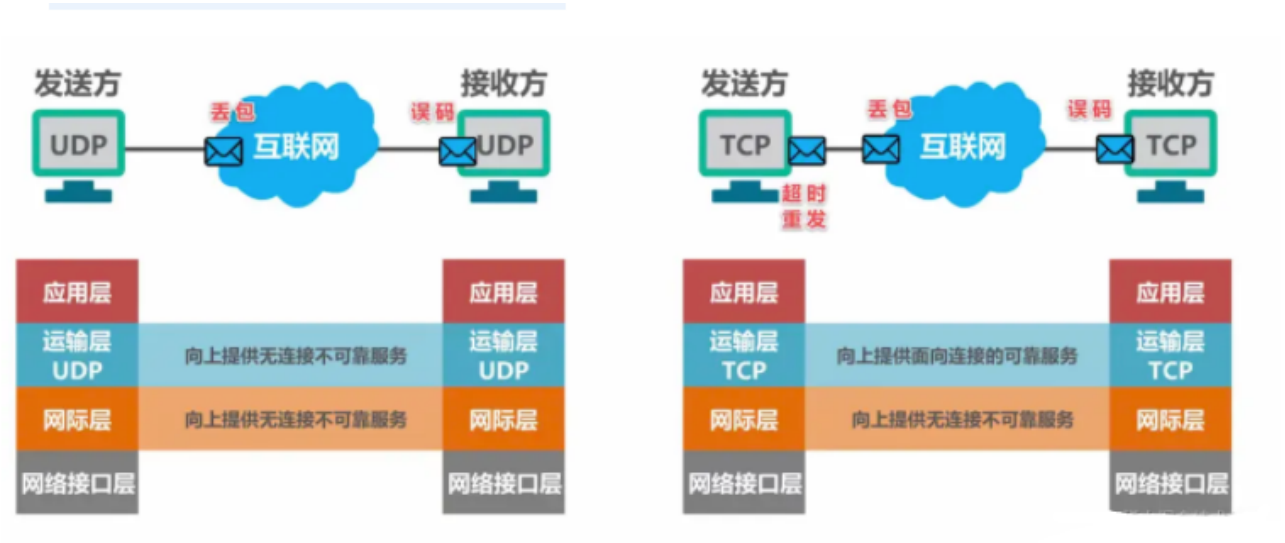
3、对应用层报文的处理

UDP：面向报文（对应用层交付的报文直接打包）
TCP：面向字节流（是tcp实现可靠传输，流量控制，拥塞控制的基础）



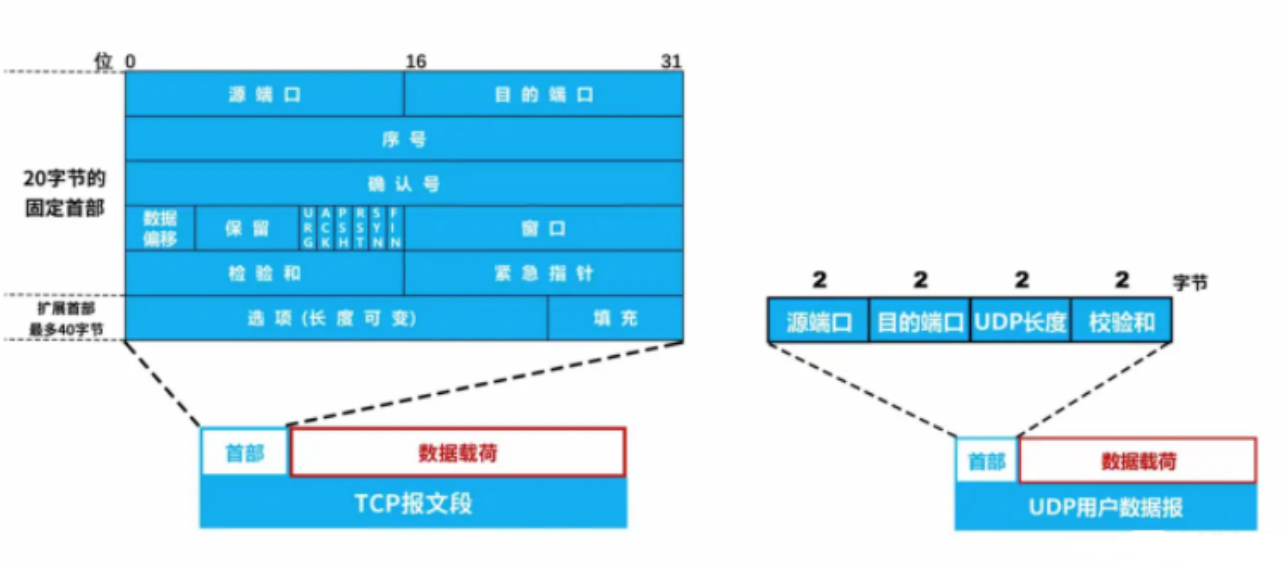
4、是否提供可靠传输

UDP：向上提供无连接不可靠服务
UDP：适用于实时应用（IP电话、视频会议等）
TCP：向上提供面向连接的可靠服务
TCP：适用于要求可靠传输的应用，例如文件传输

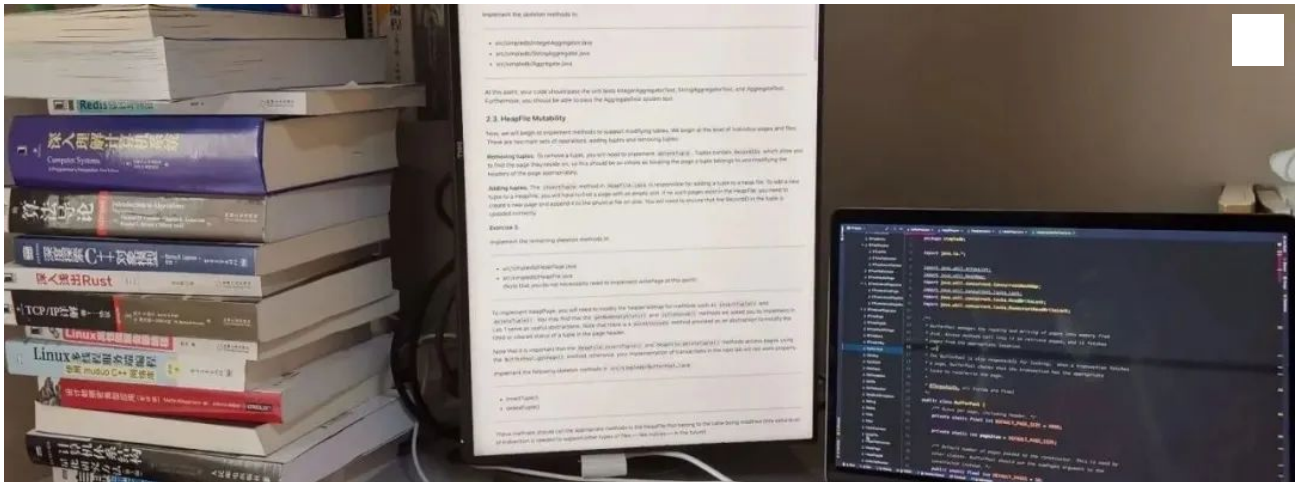


5、首部开销

UDP：8个字节
TCP：最小20字节，最大60字节



欢迎加入极客星球圈子，分享多年工作经验和基础技术深度理解，扩展视野，直播分享，面试问题，项目训练和指导，问题答疑，可以帮助想进各类大厂（芯片，自动驾驶，嵌入式，互联网等）制定学习路线和学习帮助， 可以分享各种不同公司宝贵的职场工作经验, 项目经验，普升经验，希望少走弯路，做得更好。



我的编程能力从这时候开始突飞猛进的

详细点击查看-> 极客星球