# Part Ⅰ 路由器基本功能

## MAC地址的克隆

MAC地址克隆功能可以将内网电脑的MAC地址“复制”到路由器WAN口。如使用的宽带被服务商绑定了电脑的MAC地址，要使用路由器就得需要克隆MAC地址；路由器接入公司的网络，也可能需要克隆MAC地址。

## **路由表和路由转发**

### 2.1 什么是路由

路由就是指导报文转发的路径信息，可以将一个网段的数据包转发到另外的一个网段

什么是相同的网段：相同的网络号、相同的子网掩码

路由转发涉及的表项：路由转发涉及到路由表和转发表两个表项

其中路由表是控制层面的，描述了路由的发布形式，路由接收的过程等

转发表真正指导路由转发，当数据包到达设备后直接通过转发表转发（对于华为来说转发表就是FIB表，锐捷转发表为Ref表）

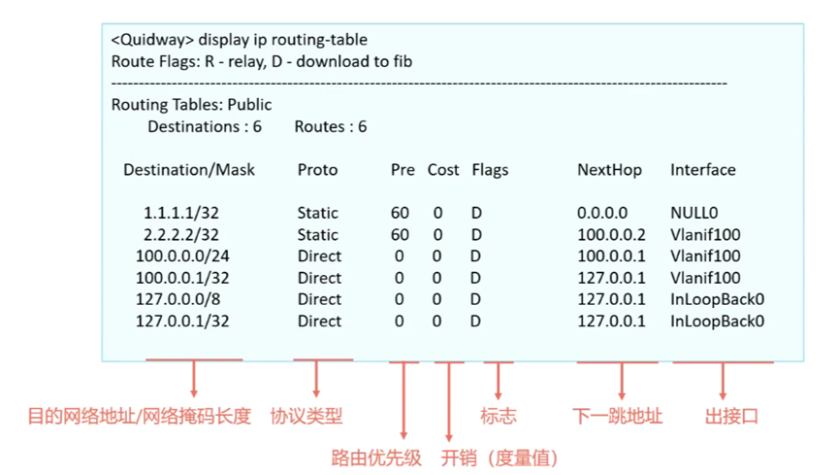
### 2.2如何实现路由转发

首先需要有具备路由功能的网络设备都来进行路由转发

该网络设备**通过路由表选择该报文匹配的路由**，然后**通过转发表进行路由转发**

路由表包含内容

路由转发主要是依靠目的网络地址/掩码 、下一跳地址、本地出接口来进行转发



**Destination/Mask**：表示此路由的目的网络地址与网络掩码。将目的地址和子网掩码“逻辑与”后可得到目的主机或路由器所在网段的地址。

例如：目的地址为1.1.1.1，掩码为255.255.255.0的主机或路由器所在网段的地址为1.1.1.0。

**Proto（Protocol）**：该路由的协议类型，也即路由器是通过什么协议获知该路由的。

**Pre（Preference）**：路由优先级（其他厂商叫管理距离 AD）

针对同一目的地，可能存在不同下一跳、出接口等多条路由，这些不同的路由可能是由不同的路由协议发现的，也可以是手工配置的静态路由。优先级最高（数值最小）者将成为当前的最优路由（有外部优先级和内部优先级）外部优先级是我们在路由表中可以看到的优先级

**路由表信息的来源**

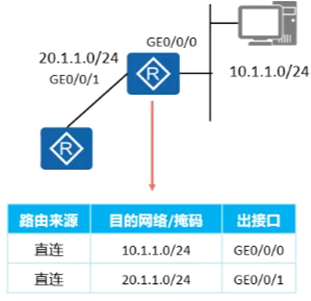
**路由信息可以通过直连、静态、动态路由学习到**

**直连路由：设备自动生成指向本地直连网络**

注意：

使用直连路由进行路由转发时，报文的目的IP和路由器接口IP在一个网段之中。

并不是所有接口生成的直连路由都会出现在路由表中，直连路由出现在路由表中的前提是该接口的物理状态、协议状态都为UP



**静态路由：管理员手工配置的路由条目**

优缺点：配置简单，对系统要求低。不能自动适应网络拓扑的变化，需要人工干预

注意：配置静态路由时可以配置下一跳，也可以配置出接口，一般对于点到点接口（串口）指定出接口

对于以太网接口和VT接口，指定下一跳（如果配置为出接口可能会存在网络不通的问题--配置为出接口表示认为该目的地址和自身是同一网段，对于以太网接口来说同网段需要封装目的IP地址的MAC，此时没有该目的IP的MAC，并且由于其实际上是不同网段的，ARP请求也得不到回应，因此无法到达该目的地址——可以通过在下一跳路由器设备开启ARP代理解决）

静态路由中有个特殊路由为缺省路由，在路由表中一0.0.0.0/0的形式存在。当报文没有在路由表中找到匹配的具体路由时，使用此路由进行转发。（一般此路由应用于企业出口）

# IMG_256

**动态路由：路由设备运行动态路由协议学习到的路由**

按照工作区域分类

IGP内部网关协议： RIPv1、RIPv2、OSPFv2、OSPFv3、IS-IS

EGP外部网关协议：BGP

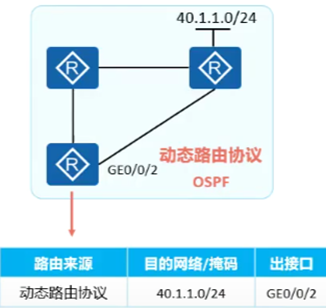
按照工作机制及算法分类

DV 距离矢量路由协议：RIPv1、RIPv2

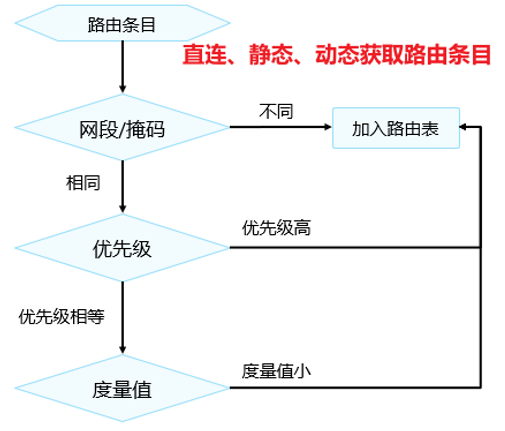
路径矢量路由协议：BGP

LS 链路状态路由协议：OSPFv2、OSPFv3、IS-IS

目前比较常用的就是BGP 、OSPF、ISIS协议



不是学到的路由都会加入到路由表中，而是从所学的路由条目中选取最优路由加入到路由表中。具体的选举规则如下（相同网段通过优先级、开销比较）



网段/掩码相同的选优先级最高的为最优路由

优先级相同的选度量值最小的为最优路由

如果网段/掩码、优先级、度量值都一致，则这些路由可以实现到达某一目的地负载分担

**路由转发的规则**

路由报文转发是通过转发表进行转发的（对于华为来说就是FIB表，锐捷为Ref表）

路由表是控制层面的，描述了路由的发布形式，路由接收的过程

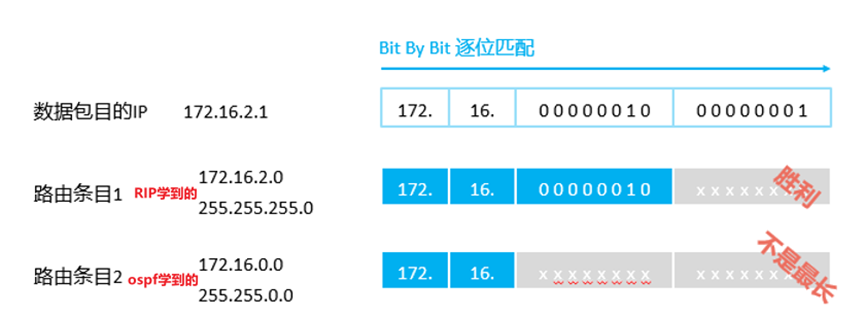
真正指导路由转发的是转发表，当数据包到达设备后直接通过转发表转发

当到达同一目的地址只有一条路由条目时，就通过此路由条目转发

当到达同一目的地址有多条路由条目时，通过最长匹配原则进行转发

最长匹配原则介绍：

当路由器收到一个IP数据包时，会将数据包的目的IP地址与自己本地路由表中的所有路由表项进行逐位（Bit-By-Bit）比对，直到找到匹配度最长的条目，然后通过此路由条目找到对应的转发表进行转发，这就是最长前缀匹配机制。

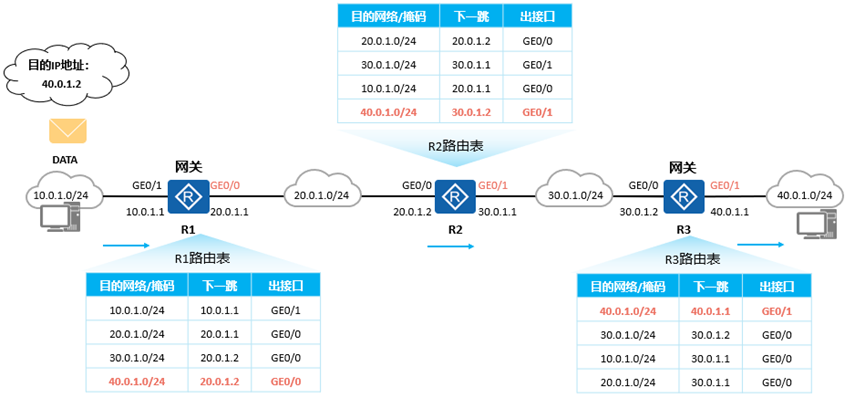


注意事项：

优先级、开销只是指导路由条目进入路由表中

路由转发时依据最长匹配原则转发，不需要路由优先级、开销这些

路由转发流程：



### 2.3 路由表和转发表的区别（不同角度解释）

1. 在路由器中，通常有两个独立的平面，通常称为“控制平面”和“转发平面”。控制平面是路由器的“大脑”，负责管理设备和路由协议等。它有时被厂商称为“路由引擎”或“监督者”。转发平面负责实际的包转发。在更大的路由器中，它通常由专用芯片和非常快的内存实现，用于路由查找。路由表(或多个路由表)驻留在控制平面中。它包含通过路由协议(OSPF，BGP，.)接收的路由信息。和从配置(连接的路由，静态路由)。如果控制平面接收到多个路由信息，则可以对给定前缀有多个条目。从所有这些信息中选择前缀的“最佳”路径。然后将此选择收集到转发表中。转发表具有确定的目的地信息，其中分组被路由到任何给定的IP前缀(或MAC地址取决于该层)。然后，这个转发表被推入路由器的转发平面，并通常分布到更大的基于底盘的路由器中的每条线上。
2. 路由表是一个L3表，它声明X.X/Y IP目的地，通过Z.Z路由器。转发表是一个L2表，用于与Z.Z路由器通信，将数据包发送到Mac地址aa:bb:cc:dd:ee:ff。在本地网络中，使用转发表获取其他主机的mac地址，并向它们发送数据包。您的网络设备将广播一个具有ip Z.Z的arp，在L2上告诉我数据包，以获取相关的mac地址。若要与不同子网中的主机通信，应通过本地网络中的路由器对其进行路由。路由表将显示该路由器具有哪个IP。使用与上面相同的转发机制到达哪个路由器。

### 2.4 跟路由有关的操作命令

#### 路由表各字段说明

Destination 目标网络或目标主机。Destination 为 default（0.0.0.0）时，表示这个是默认网关，所有数据都发到这个网关（这里是 10.139.128.1）

Gateway 网关地址，0.0.0.0 表示当前记录对应的 Destination 跟本机在同一个网段，通信时不需要经过网关

Genmask Destination 字段的网络掩码，Destination 是主机时需要设置为 255.255.255.255，是默认路由时会设置为 0.0.0.0

Flags 标记，含义参考后面的解释

Metric 路由距离，到达指定网络所需的中转数，是大型局域网和广域网设置所必需的 （不在Linux内核中使用。）

Ref 路由项引用次数 （不在Linux内核中使用。）

Use 此路由项被路由软件查找的次数

Iface 网卡名字，例如 eth0

Flags 含义：

U 路由是活动的；H 目标是个主机；G 需要经过网关；R 恢复动态路由产生的表项；D 由路由的后台程序动态地安装；M 由路由的后台程序修改；! 拒绝路由

#### Windows下对路由表的操作命令

命令参数格式：

ROUTE [-f] [-p] [-4|-6] command [destination][MASK netmask] [gateway] [METRIC metric] [IF interface]

参数解释：

-f 清除所有网关项的路由表。如果与某个命令结合使用，在运行该命令前，应清除路由表。

-p 与 ADD 命令结合使用时，将路由设置为在系统引导期间保持不变。默认情况下，重新启动系统时，不保存路由。忽略所有其他命令，这始终会影响相应的永久路由。

-4 强制使用 IPv4。

-6 强制使用 IPv6。

command 指令动作名，可取值:

PRINT 打印路由

ADD 添加路由

DELETE 删除路由

CHANGE 修改现有路由

destination 指定目标主机IP。

MASK 指定下一个参数为“netmask”值。

netmask 指定此路由项的子网掩码值。如果未指定，其默认设置为 255.255.255.255。

gateway 指定网关。

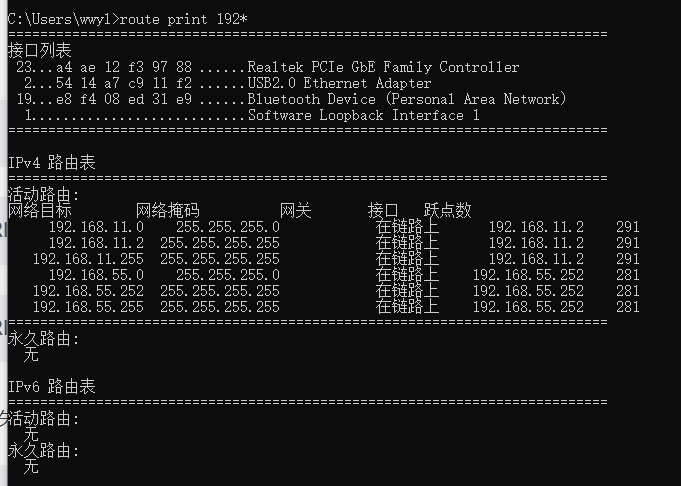
METRIC 指定跃点数，例如目标的成本。

interface 指定路由的接口号码。

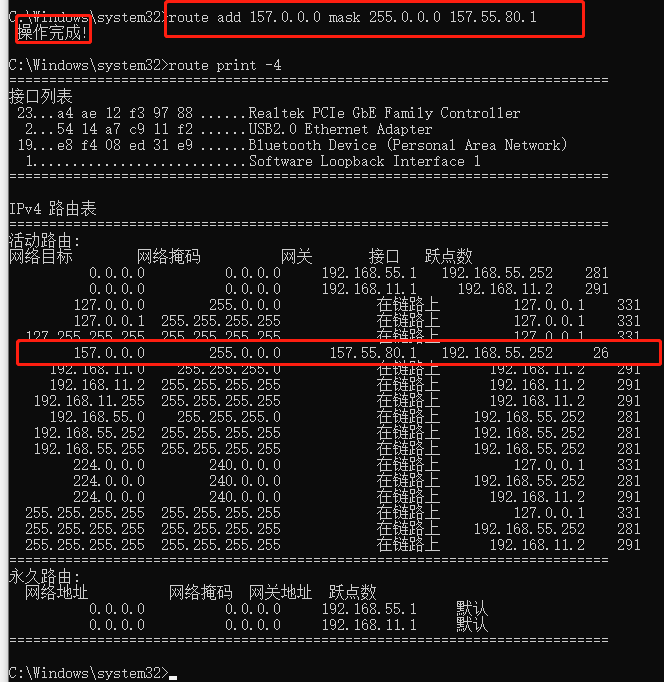
查看所有的路由表信息



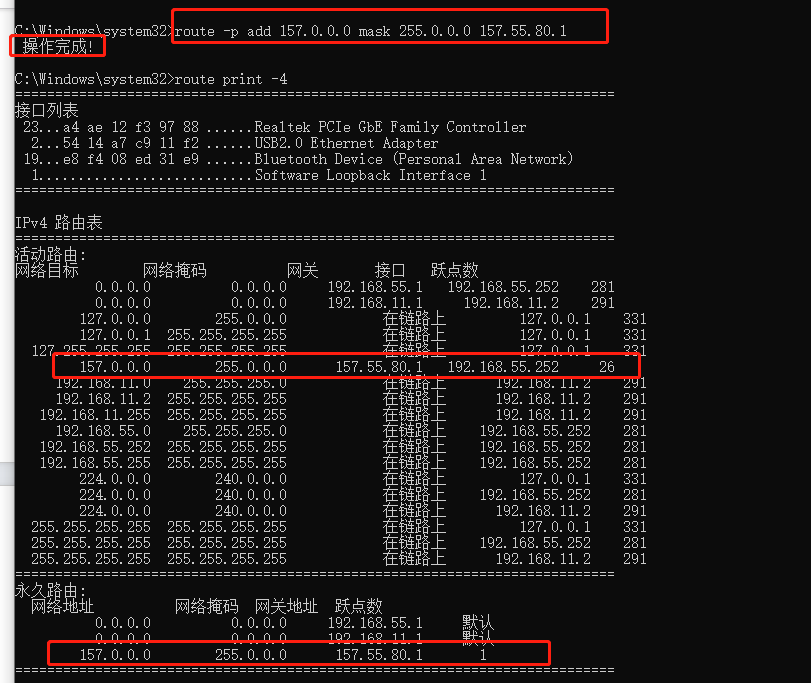
只打印那些匹配 192\* 的项



添加一条路由条目（管理员权限）



添加一条永久路由条目（-p 表示永久路由，重启后不丢失）



删除路由条目

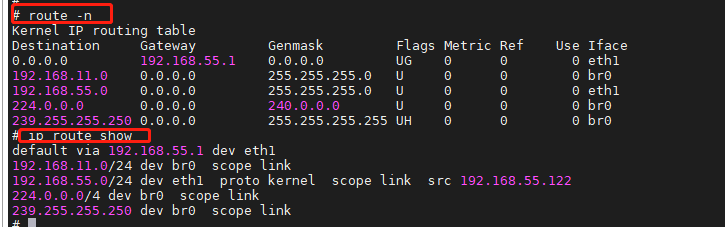


修改路由条目（CHANGE 只用于修改网关和/或跃点数）

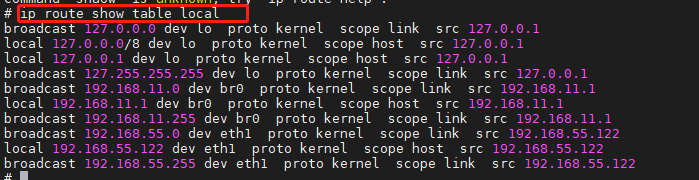
#### Linux下对路由表的操作命令：route和ip route

route是一个相当简单的工具，非常适合创建静态路由。为了兼容性，它仍然存在于许多发行版中。ip route功能更强大，功能更多，并且可以创建更专业的规则

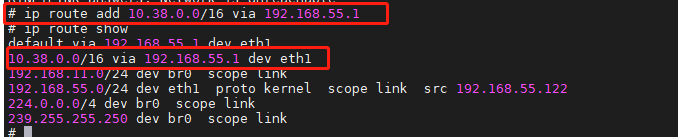
显示当前路由



查看本地路由表



添加路由



添加默认路由，修改路由等详见：http://linux-ip.net/html/tools-ip-route.html#tools-ip-route-show

# Part Ⅱ 网络基本知识

## TCP（[Transmission](https://so.csdn.net/so/search?q=Transmission&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_40431685/article/details/_blank) Control Protocol）协议

### 1.1TCP概述

传输控制协议（TCP，Transmission Control Protocol）是一种面向连接的、可靠的、基于[字节流](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82%E6%B5%81/3196772?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/TCP/_blank)的、全双工的[传输层](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82/4329536?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/TCP/_blank)[通信协议](https://baike.baidu.com/item/%E9%80%9A%E4%BF%A1%E5%8D%8F%E8%AE%AE/3351624?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/TCP/_blank)，由[IETF](https://baike.baidu.com/item/IETF/2800318?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/TCP/_blank)的[RFC](https://baike.baidu.com/item/RFC/2798645?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/TCP/_blank) 793 [1]定义。



TCP面向字节流概念(仅画出了单向)

TCP把连接作为最基本的抽象，每一条TCP连接有两个端点，即套接字(socket)或插口。如：192.168.4.5：80

套接字socket = (IP地址：端口号)

每一条TCP连接唯一的被通信两端的两个端点(即两个套接字)所确定。即：

TCP连接 ::= {socket1,socket2} = {(IP1:port1),(IP2:port2)}

### 1.2可靠传输原理及实现

可靠的传输应该满足下面两个要求：

（1）传输的信道不产生差错；

（2）保证传输数据的正确性，无差错、不丢失、不重复、并且按序到达。

这里有两层意思，一是能够正确地传输数据，二是接收方能够及时处理发送方发送的数据。

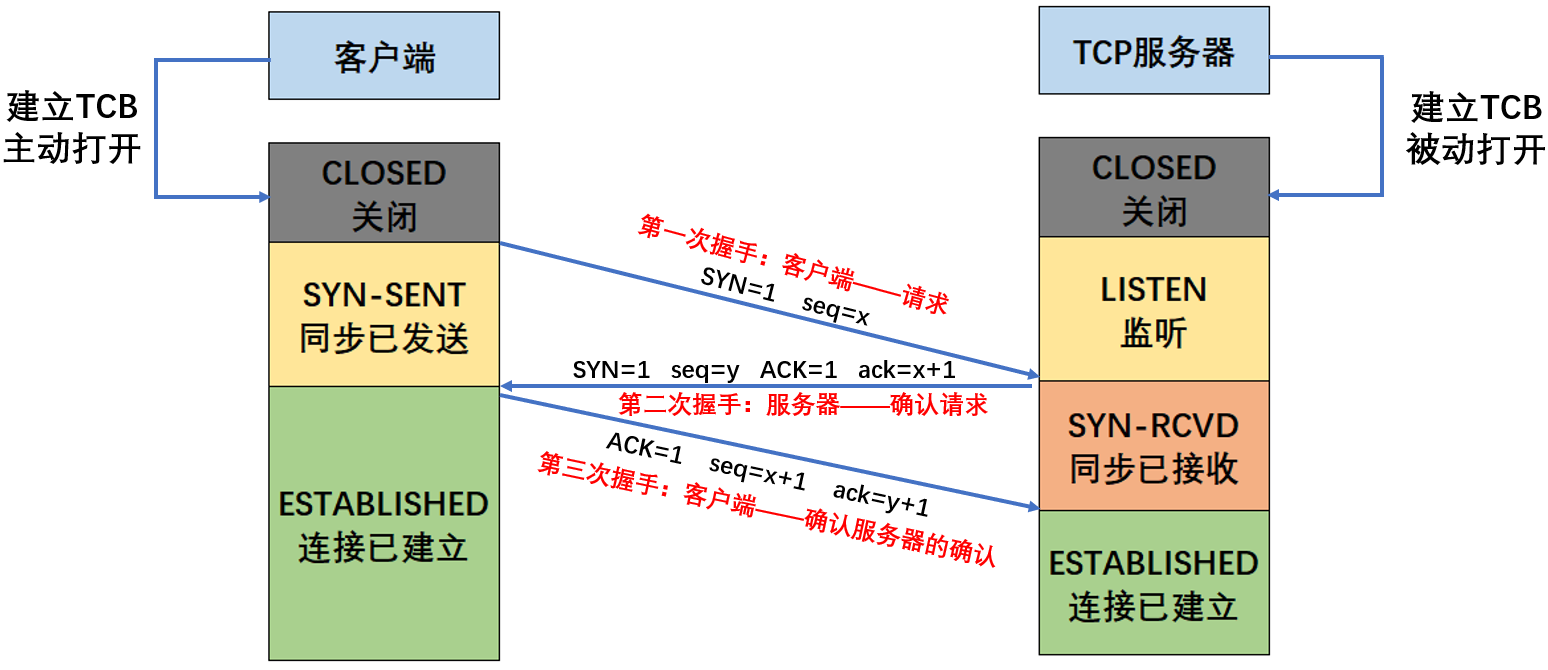
TCP提供可靠传输：

（1）采用三报文握手来建立TCP连接，四次握手来释放TCP连接，从而保证建立的传输信道是可靠的。

（2）TCP采用了连续ARQ协议（回退N，Go-back-N；超时自动重传）来保证数据传输的正确性，使用滑动窗口协议来保证接方能够及时处理所接收到的数据，进行流量控制。

（3）最后，TCP使用慢开始、拥塞避免、快重传和快恢复来进行拥塞控制，避免网络拥塞。

**三报文握手建立TCP连接**



TCB：传输控制块，打开后服务器/客户端进入监听（LISTEN）状态

SYN：TCP报文标志位，该位为1时表示发起一个新连接

ACK：TCP报文标志位，该位为1时，确认序号有效,确认接收到消息。TCP规定，在连接建立后所有报文的传输都必须把ACK置1

seq：sequence number，报文初始序列号，代表发送的第一个字节的序号

ack: acknowledgement number，报文确认序号，代表希望收到的下一个数据的第一个字节的序

**第一次握手：客户端——请求（发送请求SYN+数据包当前序列号seq，无需应答）**

客户端创建传输控制块TCB，进入监听LISTEN状态。

设置SYN=1，表示这是握手报文，并发送给服务器

设置发送的数据包序列号seq=x，此次不携带数据，但要消耗一个序列号

此时客户端处于同步已发送SYN-SENT状态

**第二次握手：服务器——确认（发送应答ACK+请求SYN+确认收到上一个数据包的确认号ack+ 当前数据包序列号seq）**

设置ACK=1，表示确认应答。

设置ack=x+1，表示已收到客户端x之前的数据，希望下次数据从x+1开始

设置SYN=1，表示握手报文，并发送给客户端

设置发送的数据包序列号seq=y，该报文同样不能携带数据，但要消耗一个序列号

此时服务器处于同步已接收SYN-RCVD状态

**第三报文握手：客户端——确认服务器的确认（发送应答ACK+确认收到上一个数据包的确认号ack+ 当前数据包序列号seq ，连接已建立，无需请求）**

设置ACK=1，表示确认应答。

设置ack=y+1，表示收到服务器发来的序列号为seq=y的包，希望下次数据从y+1开始

设置seq=x+1，表示接着上一个数据包seq=x继续发送，该报文可携带数据，但若不携带数据则不消耗序列号，下一个报文仍是seq=x+1；

至此三报文握手结束，连接建立

为什么要使用三报文握手机制？（客户端最后发的一次确认）

1.为了阻止历史连接

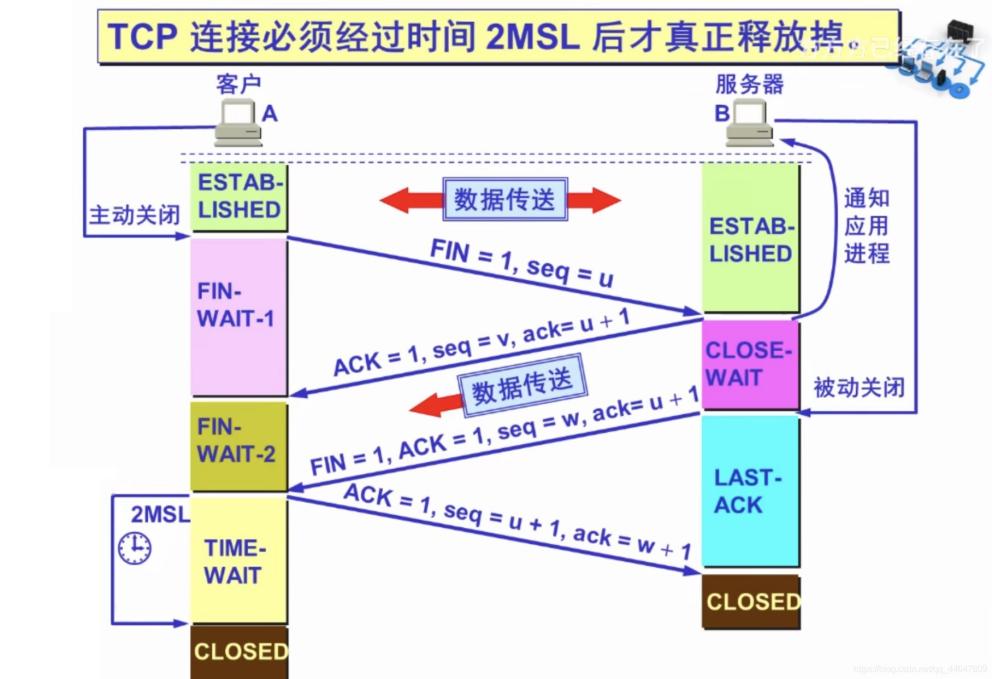
若网络状况不好的时候，客户端发送的SYN信号未能及时传输给服务器。当网络恢复时已经失效的SYN信号又到达服务器。如果只有两次握手便可建立连接，那么此时客户端就不知道这个连接是不是已经失效了的历史连接，从而导致错误的发生。三报文握手时，客户端便可以根据上下文来判断此次连接是否为历史连接，避免错误的发生。

2. 为了避免服务器开启无用连接增加服务器开销

客户端设置了一个超时时间，超过了就重新发送一个TCP连接请求。如果没有第三报文握手的话，服务端是不知道客户端是否收到服务返回的信息的，这样没有给服务器端一个创建还是关闭连接端口的请求，服务器端的端口就一直开着，等到客户端因超时重新发出请求时，服务器就会重新开启一个端口连接。那么服务器端上没有接收到请求数据的上一个端口就一直开着，长此以往，这样的端口多了，就会造成服务器端开销的严重浪费。

**四次挥手释放TCP连接**

所谓四次挥手（Four-Way Wavehand）即终止TCP连接，就是指断开一个TCP连接时，需要客户端和服务端相互总共发送4个包以确认连接的断开。在socket编程中，这一过程由客户端或服务端任一方执行close来触发。



由于TCP连接是全双工的，因此，每个方向都必须要单独进行关闭。这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了。但是在这个TCP连接上仍然能够发送数据，直到另一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，上图描述的即是如此。

**第一次挥手**：Client发送一个FIN=1，即终止控制位置为1，序号seq=u（表示：u之前的数据已经全部发送，并且数据发到u就可以截止了，就不再有数据了），用来关闭Client到Server的数据传送。Client进入FIN\_WAIT\_1（终止等待1）状态。

**第二次挥手**：Server收到FIN后，发送一个请求号seq=v和确认序号ack=u+1给Client。Server进入CLOSE\_WAIT（关闭等待）状态。

Client收到来自Server的确认后，进入FIN-WAIT-2（终止等待2状态）

Server则继续发送数据，若数据发送完成，则就开始释放连接

**第三次挥手**：Server发送一个FIN，请求号为最新的seq=w和确认序号ack=u+1，用来关闭Server到Client的数据传送。Server进入LAST\_ACK状态。

**第四次挥手**：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为w+1。Server进入CLOSED状态，完成四次挥手。

此时TCP连接还没释放掉，必须经过时间等待计时器设置的时间2MSL后才，A才进入CLOSED状态。

这最主要是因为两个理由：

1、为了保证客户端发送的最后一个ACK报文段能够到达服务器。因为这个ACK有可能丢失，从而导致处在LAST-ACK状态的服务器收不到对FIN-ACK的确认报文。服务器会超时重传这个FIN-ACK，接着客户端再重传一次确认，重新启动时间等待计时器。最后客户端和服务器都能正常的关闭。假设客户端不等待2MSL，而是在发送完ACK之后直接释放关闭，一但这个ACK丢失的话，服务器就无法正常的进入关闭连接状态。

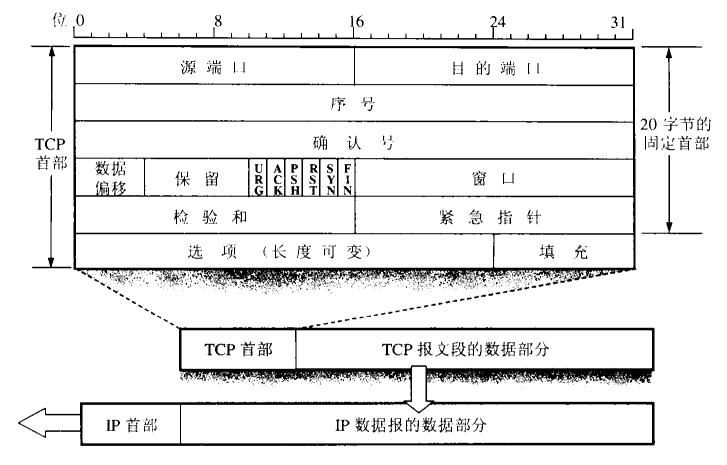
2、他还可以防止已失效的报文段。客户端在发送最后一个ACK之后，再经过经过2MSL，就可以使本链接持续时间内所产生的所有报文段都从网络中消失。从保证在关闭连接后不会有还在网络中滞留的报文段去骚扰服务器。

**TCP保活计时器**

保活计时器 (Keeplive Timer)： 目的：主要是为了防止两个TCP连接出现长时间的空闲。 当客户端与服务器端建立TCP连接后，很长时间内客户端都没有向服务器端发送数据，此时很有可能是客户端出现故障，而服务器端会一直处于等待状态。 保活计时器就是解决这种问题而生的。 工作原理：每当服务器端收到客户端的数据时，都将保活计时器重新设置（通常设置为2小时）。 过了2小时后，服务器端如果没有收到客户端的数据，会发送探测报文段给客户端，并且每隔75秒发送一个，当连续发送10次以后，仍没有收到对端的来信，则服务器端认为客户端出现故障，并会终止连接。

### 1.3 TCP报文的首部格式

TCP虽然是面向字节流的，但TCP传送的数据单元却是报文段



首部固定部分各字段的意义如下：

**源端口和目的端口**：每个TCP报文段都包含源主机和目标主机的端口号，各占2个字节，用于寻找发送端和接收端应用线程，这两个值加上IP首部中的源IP地址和目标IP地址就能确定唯一一个TCP连接。

**序号字段：**占4字节，序号范围是[0，4294967295]，序号增加到4284967295后，下一个序号就又回到0，主要是用来标识从TCP发送端向TCP接收端发送的数据字节流，它的值表示在这个报文段中的第一个数据字节所处位置，根据接收到的数据区域长度，就能计算出报文最后一个数据所处的序号，因为TCP会对发送或者接收的数据进行编号（按字节流的形式），那么使用序号对每个字节进行计数，就能很轻易管理这些数据。TCP报文段的初始序列号（ISN）是随机的，可能是0~4294967295之间的任意值.

**确认号**：包含接收端所期望收到的下一个序号，因此，确认序号应当是上次已成功收到数据的最后一个字节序号加 1。当然，只有ACK标志为 1时确认序号字段才有效，TCP为应用层提供全双工服务，这意味数据能在两个方向上独立地进行传输，因此确认序号通常会与反向数据（即接收端传输给发送端的数据）封装在同一个报文中（即捎带），所以连接的每一端都必须保持每个方向上的传输数据序号准确性。

**首部长度**：占据4bit空间（或者称数据偏移字段），它指出了TCP报文段首部长度，以字节为单位，最大能记录15\*4=60字节的首部长度，因此，TCP报文段首部最大长度为60字节。在首部长度字段后接下来有6bit空间是保留未用的。

TCP报文段首部的标志字段：

**URG**：首部中的紧急指针字段标志，如果是1表示紧急指针字段有效。

**ACK**：首部中的确认序号字段标志，如果是1表示确认序号字段有效。

**PSH**：该字段置一表示接收方应该尽快将这个报文段交给应用层。

**RST**：重新建立TCP连接。

**SYN**：用同步序号发起连接。

**FIN**：中止连接。

TCP的流量控制由连接的每一端通过声明的窗口大小来提供，窗口大小为字节数，起始于确认序号字段指明的值，这个值是接收端正期望接收的数据序号，发送方根据窗口大小调整发送数据，以实现流量控制。窗口大小是一个占据16 bit空间的字段，因而窗口最大为 65535字节，当接收方告诉发送方一个大小为0的窗口时，将完全阻止发送方的数据发送。

只有当URG标志置1时紧急指针才有效，紧急指针是一个正的偏移量，和序号字段中的值相加表示紧急数据最后一个字节的序号。简单来说，本TCP报文段的紧急数据在报文段数据区域中，从序号字段开始，偏移紧急指针的值结束。

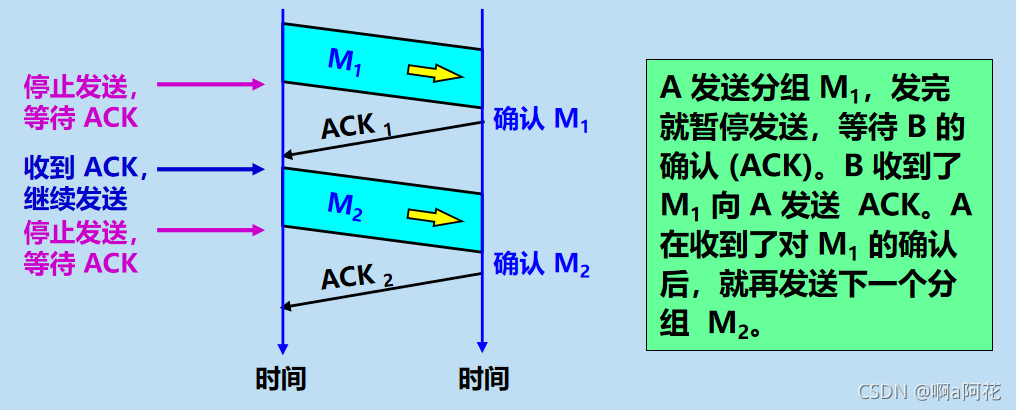
### 1.4 ARQ协议和滑动窗口（流量控制，点对点，只和主机有关）

ARQ协议，即自动重传请求（Automatic Repeat-reQuest），是OSI模型中的错误纠正协议之一；它通过使用确认和重传这两个机制，在不可靠服务的基础上实现可靠的信息传输；如果发送方在发送后一段时间之内没有收到确认帧，它通常会重新发送；重传的请求是自动进行的，接收方不需要请求发送方重传某个出错的分组；ARQ包括停止等待ARQ协议和连续ARQ协议

**停止等待ARQ**

（1）正常传输

停止等待”就是每发送完一个分组就停止发送，等待对方的确认。在收到确认后再发送下一个分组。全双工通信的双方既是发送方也是接收方。为了讨论问题的方便，仅考虑 A 发送数据，而 B 接收数据并发送确认。因此 A 叫做发送方，而 B 叫做接收方

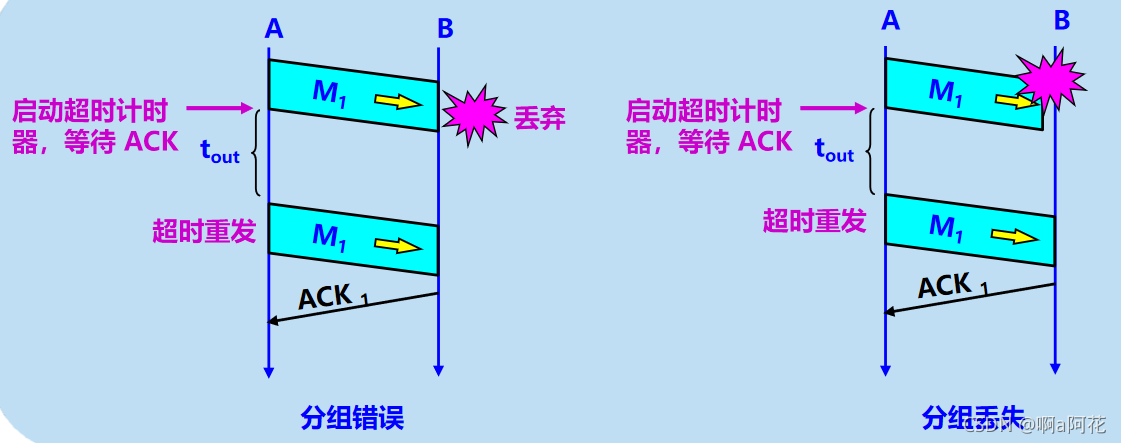


（2）传输出现差错

在接收方 B 会出现两种情况：

1. B 接收 M1 时检测出了差错，就丢弃 M1，其他什么也不做（不通知 A 收到有差错的分组）。
2. M1 在传输过程中丢失了，这时 B 当然什么都不知道，也什么都不做。

在这两种情况下，B 都不会发送任何信息。A都必须重发分组，直到B正确接收为止，这样才能实现可靠通信。



**超时重传**

A怎么知道 B 是否正确收到了 M1 呢？A 为每一个已发送的分组都设置了一个超时计时器。A 只要在超时计时器到期之前收到了相应的确认，就撤销该超时计时器，继续发送下一个分组 M2 。

**丢失确认**

若A在超时计时器规定时间内没有收到B的确认ACK，就重发该分组。无论是自己发送的分组出错、丢失了，或者 是 B 发送的确认丢失了。

假定 B 又收到了重传的分组 M1。这时 B 应采取两个行动：

1、丢弃这个重复的分组 M1，不向上层交付。

2、向 A 发送确认。不能认为已经发送过确认就不再发送，因为 A 之所以重传 M1 就表示 A 没有收到对 M1 的确认

**编号**

若分组正确到达B，但B回送的确认ACK丢失或延迟了，A未收到B的确认，会超时重发。B 可能会收到重复的 M1 。B如何知道收到了重复的分组，需要丢弃呢？

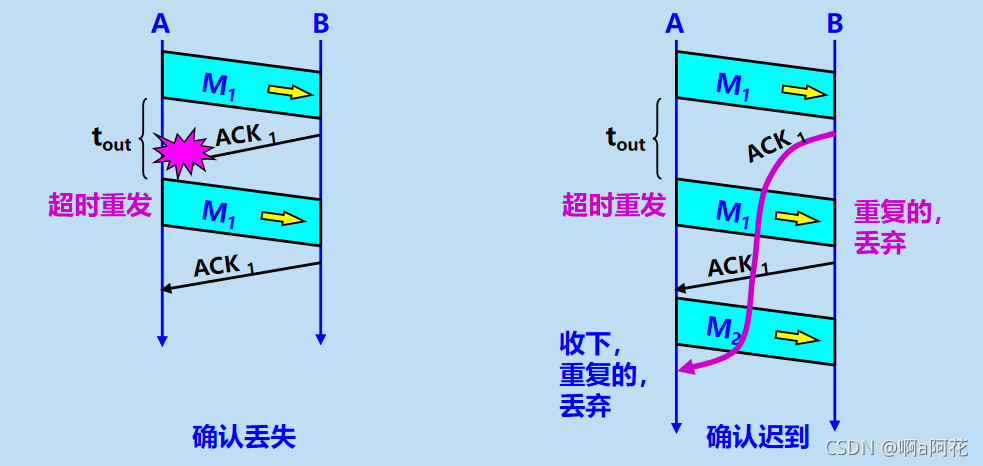
A为每一个发送的分组都进行编号。若B收到了编号相同的分组，则认为收到了重复分组，丢弃重复的分组，并回送确认（之所以会收到重复的分组就是因为发送端没有正常收到确认）。

B为发送的确认也进行编号，指示该确认是对哪一个分组的确认。A根据确认及其编号，可以确定它是对哪一个分组的确认，避免重复发送。若为重复的确认，则将其丢弃。

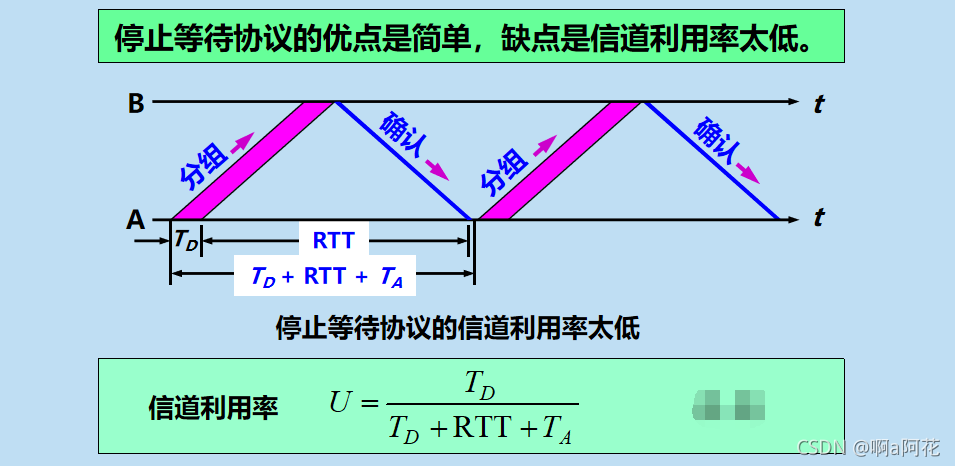
**迟到确认**

传输过程中没有出现差错，但 B 对分组 M1 的确认迟到了。A会超时重发M1， 则B 收到重复的 M1，同样要丢弃重复的 M1，并重传确认分组。

稍后B迟到的确认和重发的确认都会到达A，A收到重复的确认，对重复的确认的处理很简单：收下后就丢弃。



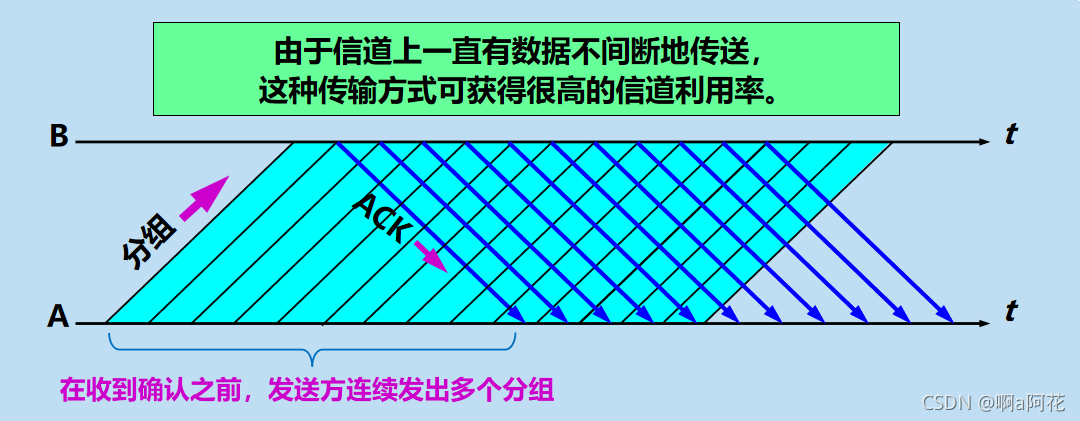
1. 信道利用率



**连续ARQ**

（1）流水线传输

为了提高传输效率，发送方可连续发送多个分组，不必每发完一个分组就停顿下来等待对方的确认。这样可使信道上一直有数据不间断地传送。



流水线传输通常要求

1. 必须增加序号范围，因为每个输送中的分组（不计算重传的）必须有一个唯一的序号，因为可能存在有多个在输送中未确认的报文。
2. 协议的发送方和接收方要能缓存多个分组。发送方至少要能缓存那些已发送但没有确认的分组，接收方可能要缓存那些已正确接收的分组。
3. 所需序号范围和对缓冲的要求取决于数据传输协议如何处理丢失、损坏及延时过大的分组。
4. 解决流水线传输的差错有两种滑动窗口协议：回退N(Go-Back- N，GBN) 和选择重传(Selective Repeat，SR) 。

**累积确认**

即接收方不必对收到的分组逐个发送确认，而是对按序到达的最后一个分组发送确认。

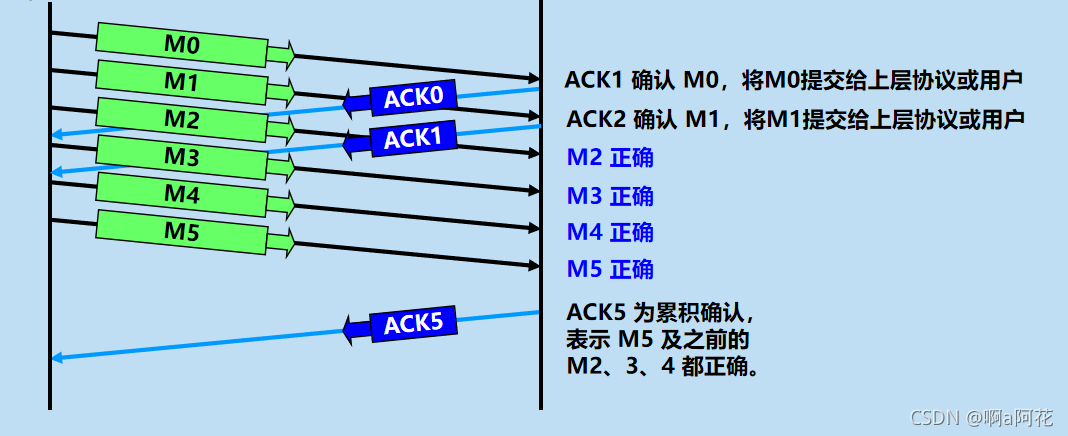
因此，接收方不必对中间某个确认包的丢失进行重传，因为累积确认的效果，一旦一个较大序号的ACK被接收到，则表示这个包之前的包都已经成功收到。

如果接受方收到了这个确认，则表示到这个分组为止之前的所有分组都已正确收到了。

但同时累积确认也有缺陷，例如发送方传送了5个包，但是第3个包丢失了，第4和第5个正常。这时根据累积确认，接收方只能传送前两个包的ACK（按序到达的最后一个）。在接收到第2个包ACK的发送方不知道后面3个包是否到达，就会将后3个包全部进行重传。即：Go-back-N。

优点：容易实现，即使部分确认丢失也不必重传。

缺点：不能向发送方反映出接收方已经正确收到的所有分组的信息。



注：TCP并不是每一个报文段都会回复ACK的，可能会对一个报文段发送一个ACK（M0、M2），也可能会对多个报文段发送1个ACK（M3+M4+M5）

（2）滑动窗口

滑动窗口协议（Sliding Window Protocol）以基于分组的数据传输协议为特征。因此该协议适用于对按顺序传送分组的可靠性要求较高的环境，例如在数据链路层（OSI模型）以及传输控制协议（TCP）中。

* 提供TCP的可靠性（最基本的传输可靠性来源于“确认重传”机制），避免丢包
* 提供TCP的流控特性，用于网络数据传输时的流量控制，以避免拥塞的发生。

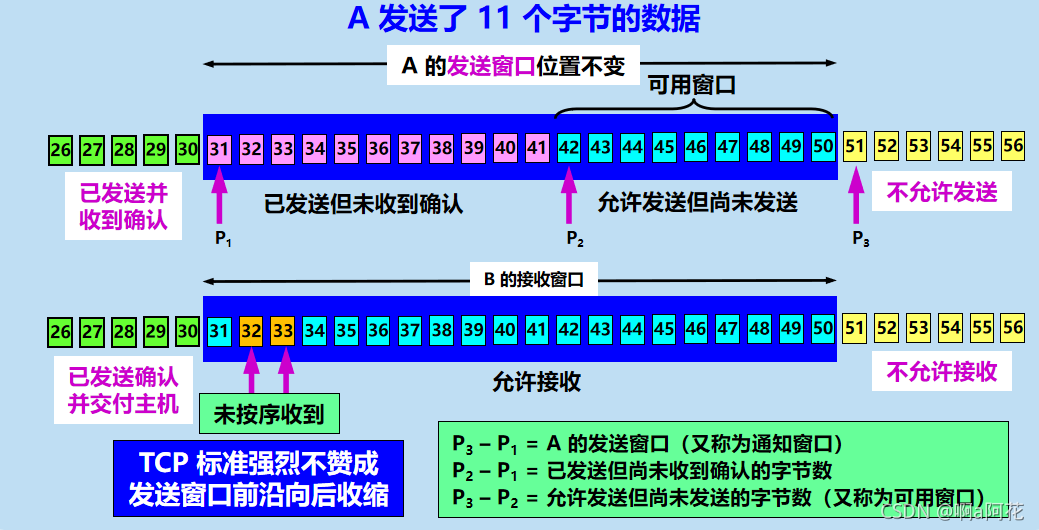
1. 滑动窗口协议在发送方和接收方之间各自维持一个滑动窗口，发送方是发送窗口，接收方是接收窗口（两个窗口大小不一定相同）
2. 使用滑动窗口协议控制发送方和接收方所能发送和接收的分组的数量和编号
3. 它允许发送方发送多个分组而不需等待确认，TCP的滑动窗口是以字节为单位的
4. 每收到一个确认，发送方就把发送窗口向前滑动。

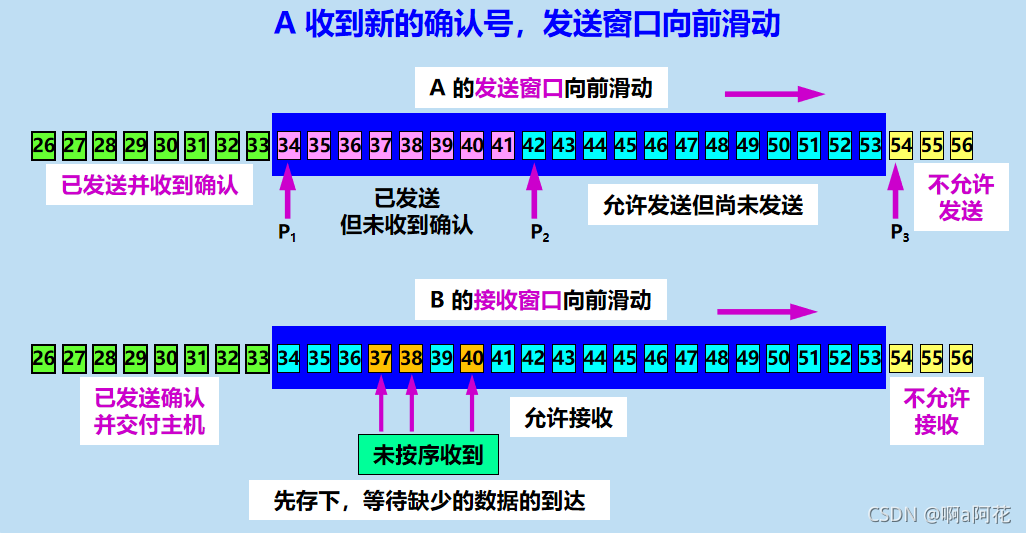
发送窗口中有四个概念：

* 已发送并收到确认的数据（不在发送窗口和发送缓冲区之内）
* 已发送但未收到确认的数据（位于发送窗口之内）
* 允许发送但尚未发送的数据（位于发送窗口之内）
* 发送窗口之外的缓冲区内暂时不允许发送的数据

接收窗口中也有四个概念：

* 已发送确认并交付主机的数据（不在接收窗口和接收缓冲区之内）
* 未按序收到的数据（位于接收窗口之内）
* 允许的数据（位于接收窗口之内）
* 不允许接收的数据（位于发送窗口之内）





当A 的发送窗口内的序号都已用完，但还没有再收到确认时，必须停止发送

为了解决流水线传输的差错，TCP协议提供两种滑动窗口：回退N(Go-Back- N，GBN) 和选择重传(Selective Repeat，SR) 来解决这个问题

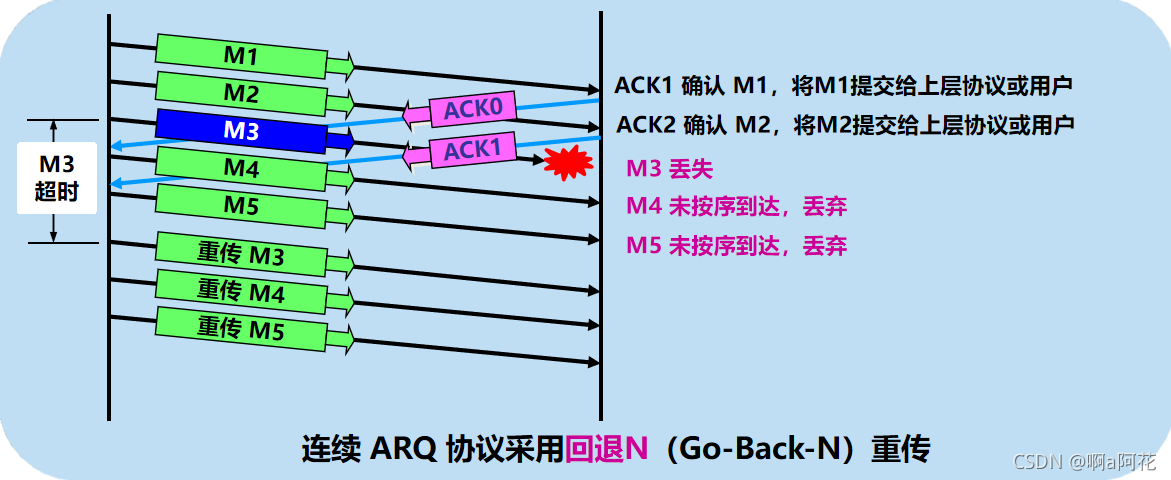
1. 回退N帧

如果发送方发送了前 5 个分组，而中间的第 3 个分组丢失了。这时接收方只能对前两个分组发出确认。

发送方无法知道后面三个分组的下落，而只好把后面的三个分组都再重传一次。

这就叫做 Go-back-N（回退 N），表示需要再退回来重传已发送过的 N 个分组。

可见当通信线路质量不好时，连续 ARQ 协议会带来负面的影响



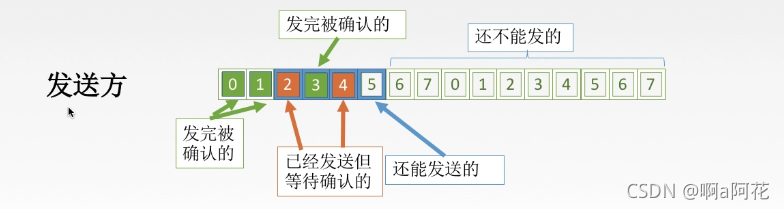
如果下面的网络层协议丢失了很多分组,那么返回N协议的效率就会很低。

因为每当一个分组损坏,发送方就需要重传所有待确认的分组,虽然其中有些分组实际上已经完整地被接收了

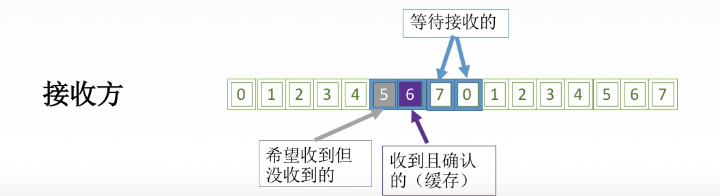
2. 选择重传

选择重传协议只重传真正出错或者丢失的帧，那怎么实现呢？

发送方维持着一个窗口，包含可发送或已发送但未被确认的序号。



接收方维持着一个窗口，包含可接收的序号，每个序号还保留一个缓冲区。与每个缓冲区项相关联的还有一个标志位，用来指明缓冲区是满的还是空的。



每到达一个帧，接收方通过检查它的序号，看是否落在窗口内。如果确实落在窗口内且之前没有接收过，则接收该分组然后保存在缓冲区并返回一个确认。等前面这些帧都到达后一起交付给上层

发送端每个发送缓冲都设有一个超时一个超时计时器，一个超时事件发生后相应缓冲区的帧就会被重发

**停止等待ARQ和连续ARQ比较**



注意： 在发送完一个分组后，必须暂时保留已发送的分组的副本，以备重发。

分组和确认分组都必须进行编号。

超时计时器的重传时间应当比数据在分组传输的平均往返时间更长一些。

**ARQ到底运行在那一层？**

ARQ是一种可以在不可靠的数据通道上可靠地传输数据的方案，所以其实链路层和传输层都用了ARQ，并不专属某一层。

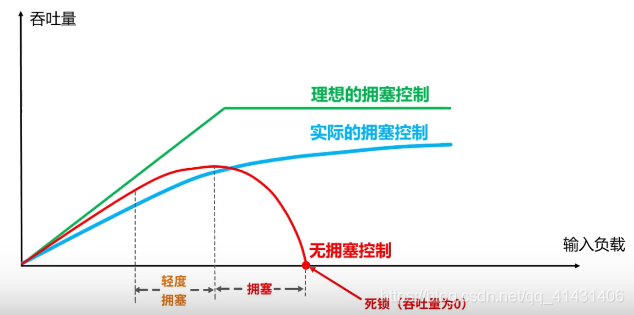
并不是一条连接只要有一层用了ARQ，它的上层的通信就是可靠的。因为ARQ只保证使用它的点到点是可靠的，比如数据链路层只保证你和你的路由器通信可靠，你的路由器到小区的路由器通信也可靠， 但是路由器本身会故障，会拥塞丢包，也就是点本身会产生问题。

所以需要在传输层或者应用层再加一层ARQ保障整条数据通道的可靠性。比如你自己写程序要在应用层通信，但传输层不用tcp想用udp，也可以在你程序里用ARQ协来实现可靠性。

### 1.5 TCP拥塞控制（全局）

什么叫网络拥塞？

在某段时间，若对网络中某一资源的需求超过了该资源所能提供的可用部分，网络性能就要变坏，这种情况就叫做网络拥塞。在计算机网络中数位链路容量（即带宽）、交换结点中的缓存和处理机等，都是网络的资源。若出现拥塞而不进行控制，整个网络的吞吐量将随输入负荷的增大而下降。



TCP的四种拥塞控制算法

1. 慢开始
2. 拥塞控制
3. 快重传
4. 快恢复

假定：

1.数据是单方向传送，而另一个方向只传送确认

2.接收方总是有足够大的缓存空间，因而发送发发送窗口的大小由网络的拥塞程度来决定

3.以TCP报文段的个数为讨论问题的单位，而不是以字节为单位



示例如下：

传输轮次：发送方给接收方发送数据报文段后，接收方给发送方发回相应的确认报文段，一个传输轮次所经历的时间就是往返时间RTT(RTT并非是恒定的数值），使用传输轮次是为了强调，把拥塞窗口cwnd所允许发送的报文段都连续发送出去，并收到了对已发送的最后一个报文段的确认，拥塞窗口cwnd会随着网络拥塞程度以及所使用的拥塞控制算法动态变化。

在tcp双方建立逻辑链接关系时， 拥塞窗口cwnd的值被设置为1，还需设置慢开始门限ssthresh,在执行慢开始算法时，发送方每收到一个对新报文段的确认时，就把拥塞窗口cwnd的值加一，然后开始下一轮的传输，当拥塞窗口cwnd增长到慢开始门限值时，就使用拥塞避免算法。

慢开始：

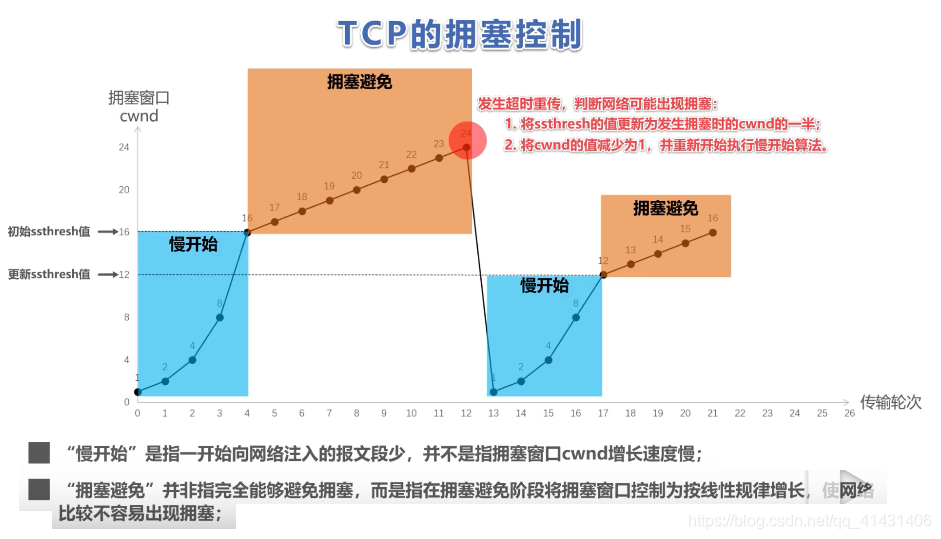
假设当前发送方拥塞窗口cwnd的值为1，而发送窗口swnd等于拥塞窗口cwnd，因此发送方当前只能发送一个数据报文段（拥塞窗口cwnd的值是几，就能发送几个数据报文段），接收方收到该数据报文段后，给发送方回复一个确认报文段，发送方收到该确认报文后，将拥塞窗口的值变为2，

发送方此时可以连续发送两个数据报文段，接收方收到该数据报文段后，给发送方一次发回2个确认报文段，发送方收到这两个确认报文后，将拥塞窗口的值加2变为4，发送方此时可连续发送4个报文段，接收方收到4个报文段后，给发送方依次回复4个确认报文，发送方收到确认报文后，将拥塞窗口加4，置为8，发送方此时可以连续发送8个数据报文段，接收方收到该8个数据报文段后，给发送方一次发回8个确认报文段，发送方收到这8个确认报文后，将拥塞窗口的值加8变为16，

当前的拥塞窗口cwnd的值已经等于慢开始门限值，之后改用拥塞避免算法。

拥塞避免：

也就是每个传输轮次，拥塞窗口cwnd只能线性加一，而不是像慢开始算法时，每个传输轮次，拥塞窗口cwnd按指数增长。同理，16+1……直至到达24，假设24个报文段在传输过程中丢失4个，接收方只收到20个报文段，给发送方依次回复20个确认报文段，一段时间后，丢失的4个报文段的重传计时器超时了，发送发判断可能出现拥塞，更改cwnd和ssthresh.并重新开始慢开始算法，如图所示：

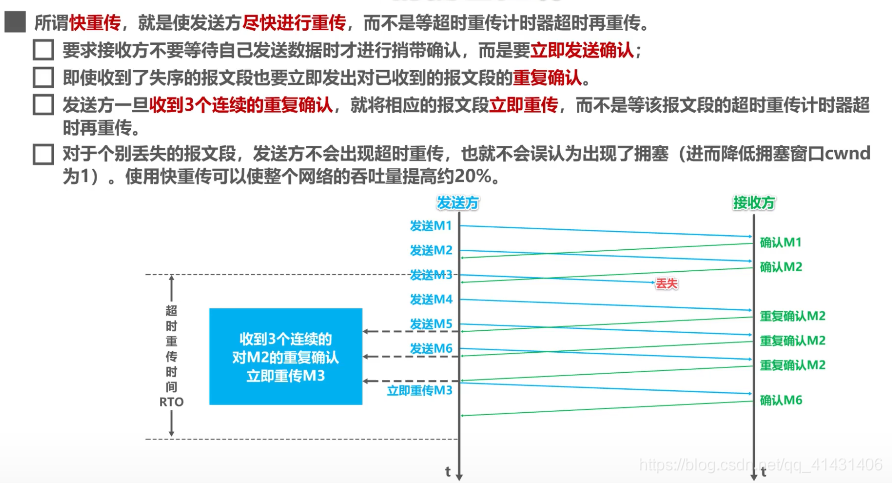


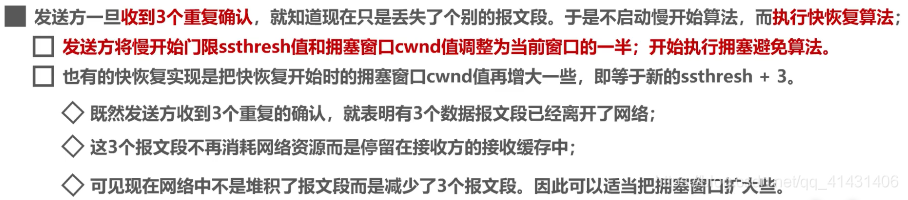
快速重传：

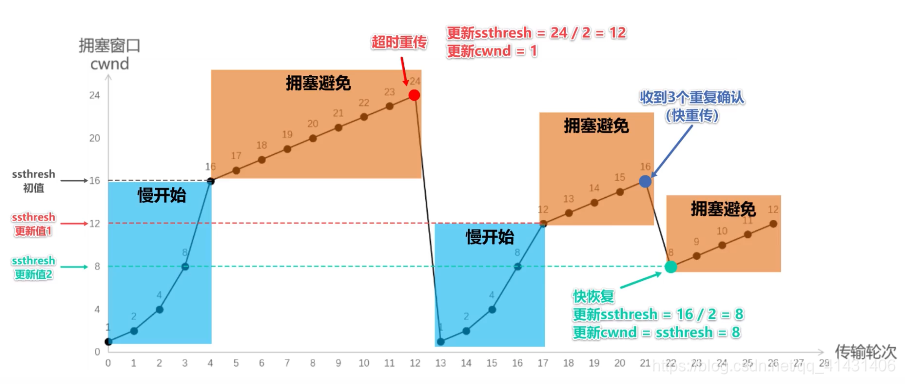
发送方发送1号数据报文段，接收方收到1号报文段后给发送方发回对1号报文段的确认，在1号报文段到达发送方之前，发送方还可以将发送窗口内的2号数据报文段发送出去，接收方收到2号报文段后给发送方发回对2号报文段的确认，在2号报文段到达发送方之前，发送方还可以将发送窗口内的3号数据报文段发送出去，

假设该报文丢失，发送方便不会发送针对该报文的确认报文给发送方，发送方还可以将发送窗口内的4号数据报文段发送出去，接收方收到后，发现这不是按序到达的报文段，因此给发送方发送针对2号报文段的重复确认，表明我现在希望收到的是3号报文段，但是我没有收到3号报文段，而收到了未按序到达的报文段，发送方还可以将发送窗口中的5号报文段发送出去,接收方收到后，发现这不是按序到达的报文段，因此给发送方发送针对2号报文段的重复确认，表明我现在希望收到的是3号报文段，但是我没有收到3号报文段，而收到了未按序到达的报文段,，发送方还可以将发送窗口内的最后一个数据段即6号数据报文段发送出去，接收方收到后，发现这不是按序到达的报文段，因此给发送方发送针对2号报文段的重复确认，表明我现在希望收到的是3号报文段，但是我没有收到3号报文段，而收到了未按序到达的报文段，

此时，发送方收到了累计3个连续的针对2号报文段的重复确认，立即重传3号报文段，接收方收到后，给发送方发回针对6号报文的确认，表明，序号到6为至的报文都收到了，这样就不会造成发送方对3号报文的超时重传，而是提早收到了重传。







### 1.6 三报文握手和四次挥手抓包

#### 三报文握手---建立TCP连接

目标：访问http://www.sina.com.cn，抓取本机与目标主机通信时的TCP三报文握手建立连接报文

步骤

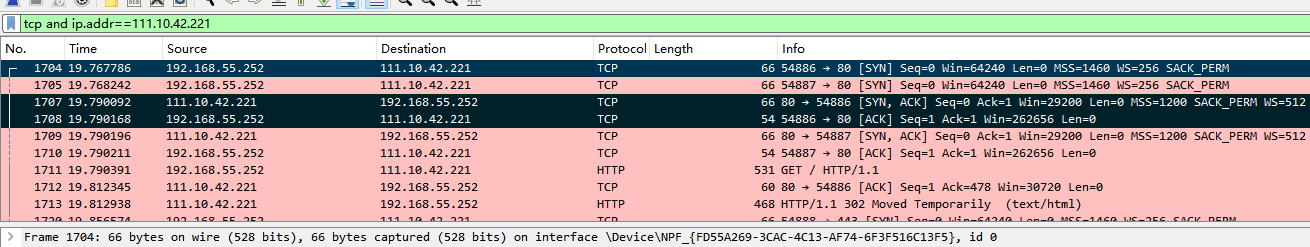
①wireshark开启抓包

②访问http://www.sina.com.cn

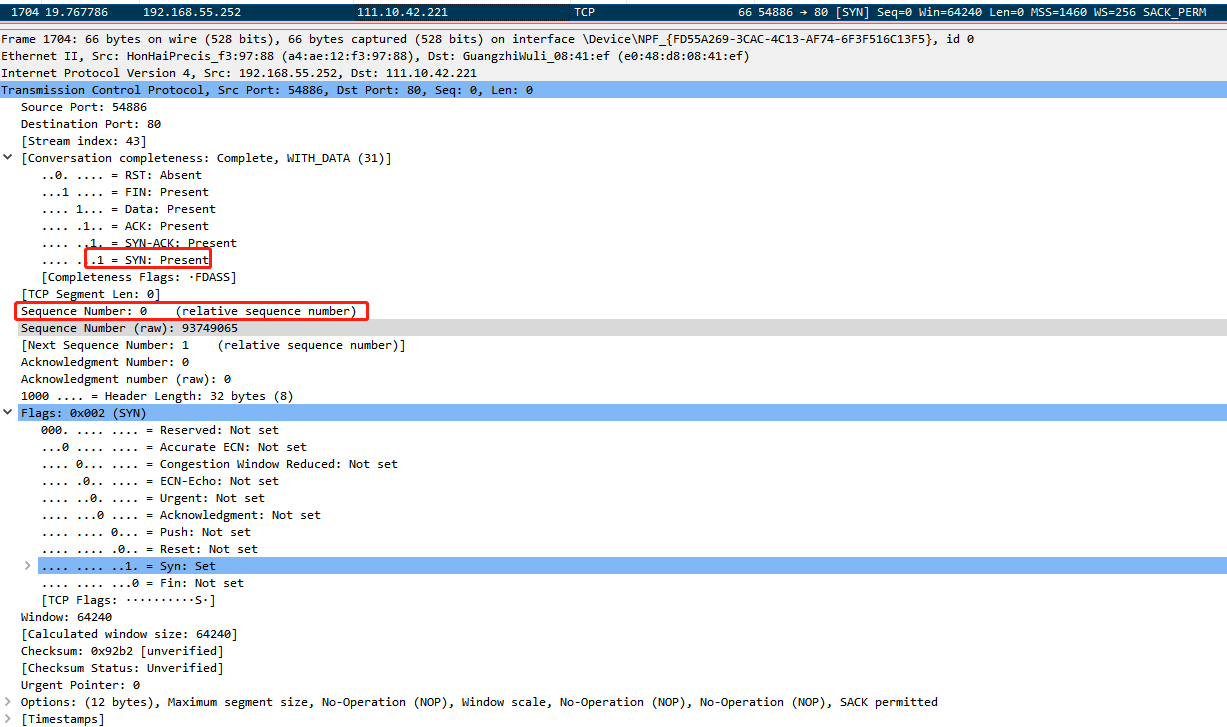
③通过ping命令找到目标域名对应的目的ip

④筛选目的ip的tcp协议包

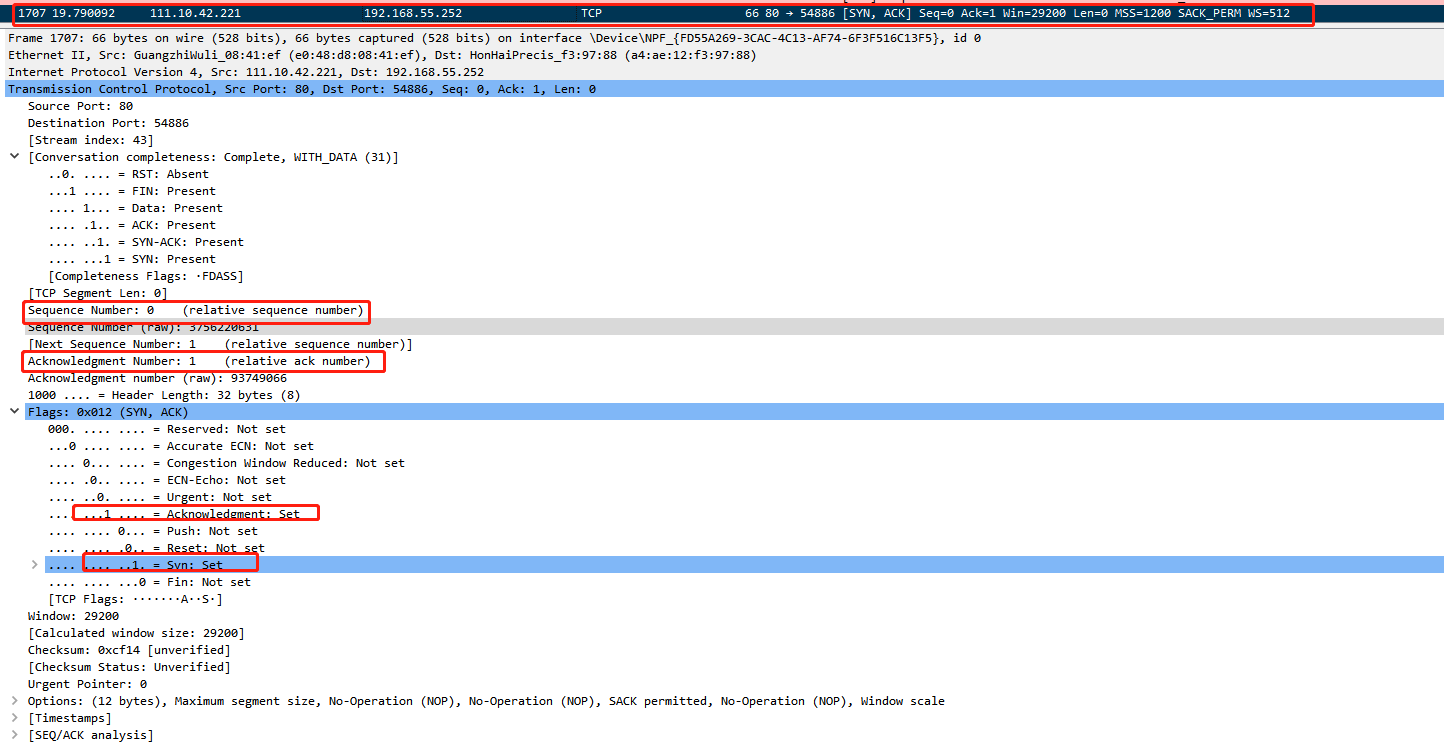
三报文握手：标记部分



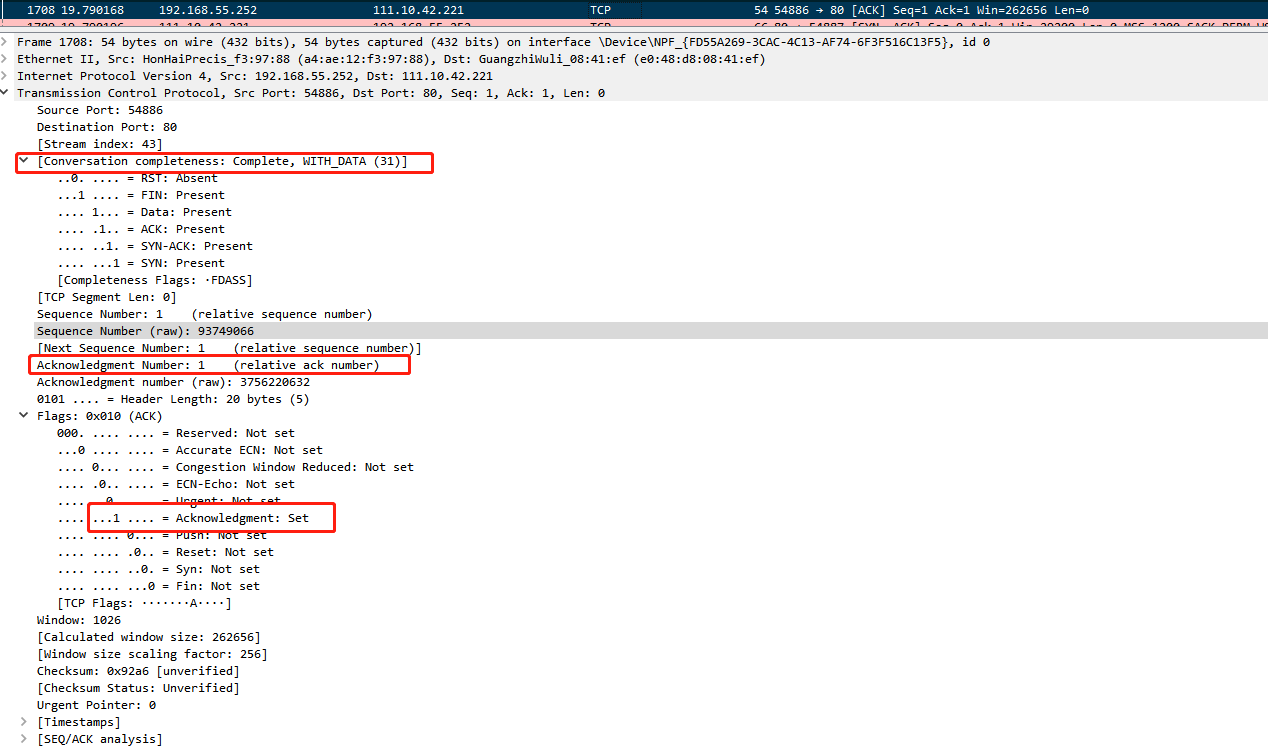
第一次



第二次



第三次



#### 四次挥手---释放连接

目标：访问http://dl.xmjlwl.cn，抓取本机与目标主机通信时的TCP四次挥手释放连接报文

步骤

①wireshark开启抓包

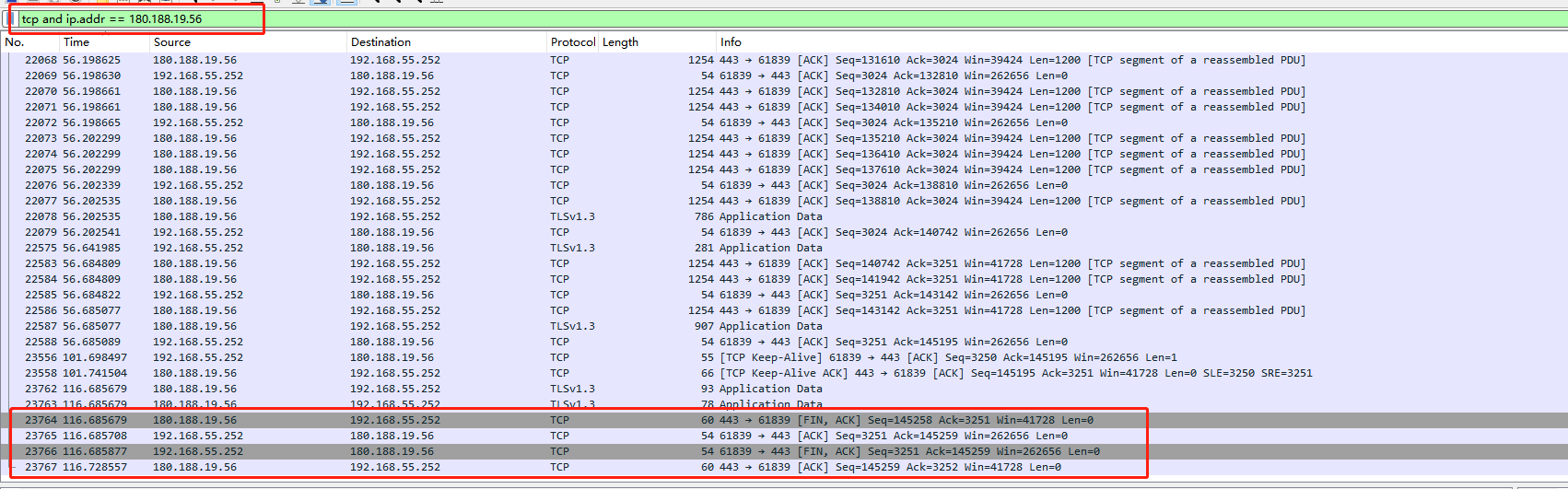
②访问http://dl.xmjlwl.cn

③通过ping命令找到目标域名对应的目的ip

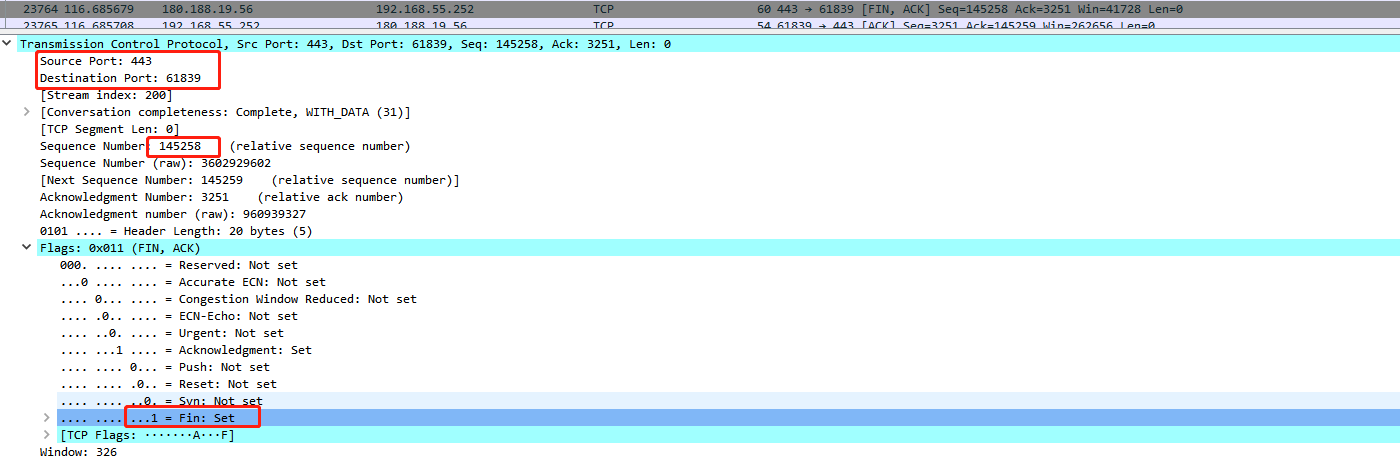
④筛选目的ip的tcp协议包

注意：TCP 连接的任意一方都可以发起关闭操作，只不过通常情况下发起关闭连接操作一般都是客户端。 然而，一些服务器比如 Web 服务器在对请求作出相应后也会发起关闭连接的操作。 TCP 协议规定通过发送一个 FIN 报文来发起关闭操作。

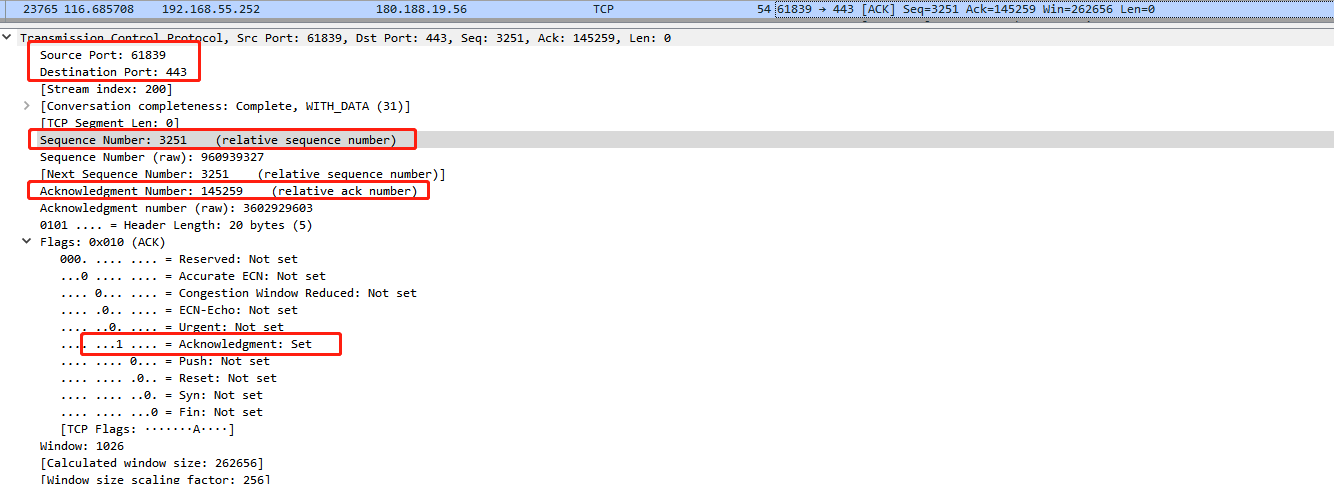
四次挥手：标记部分



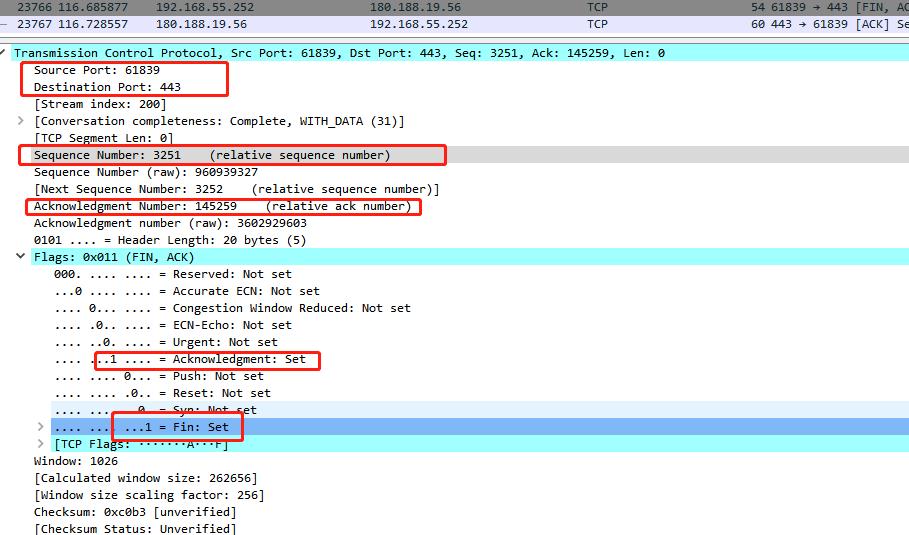
第一次挥手：由服务端180.188.19.56(dl.xmjlwl.cn)发起断开连接请求，FIN=1，seq=145258



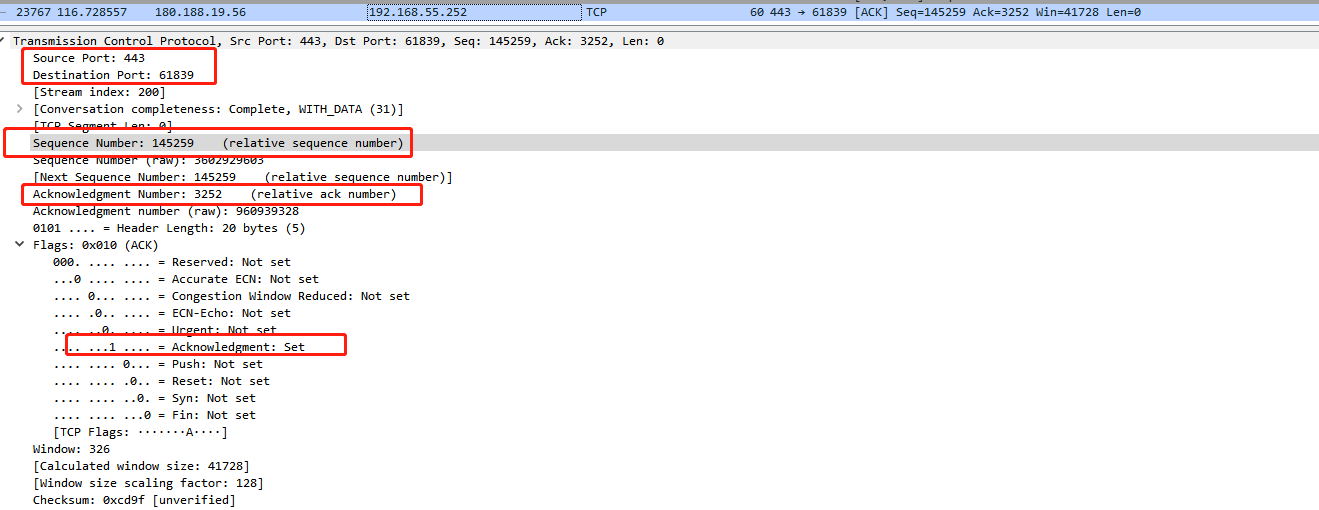
第二次挥手：由客户端：192.168.55.252回应，ACK=1，seq=3251，ack=145259



第三次挥手：由客户端192.168.55.252发起关闭连接，FIN=1，ACK=1，seq= 3251，ack= 145259



第四次挥手：由服务端180.188.19.56回应，ACK=1，seq=145259，ack=3252



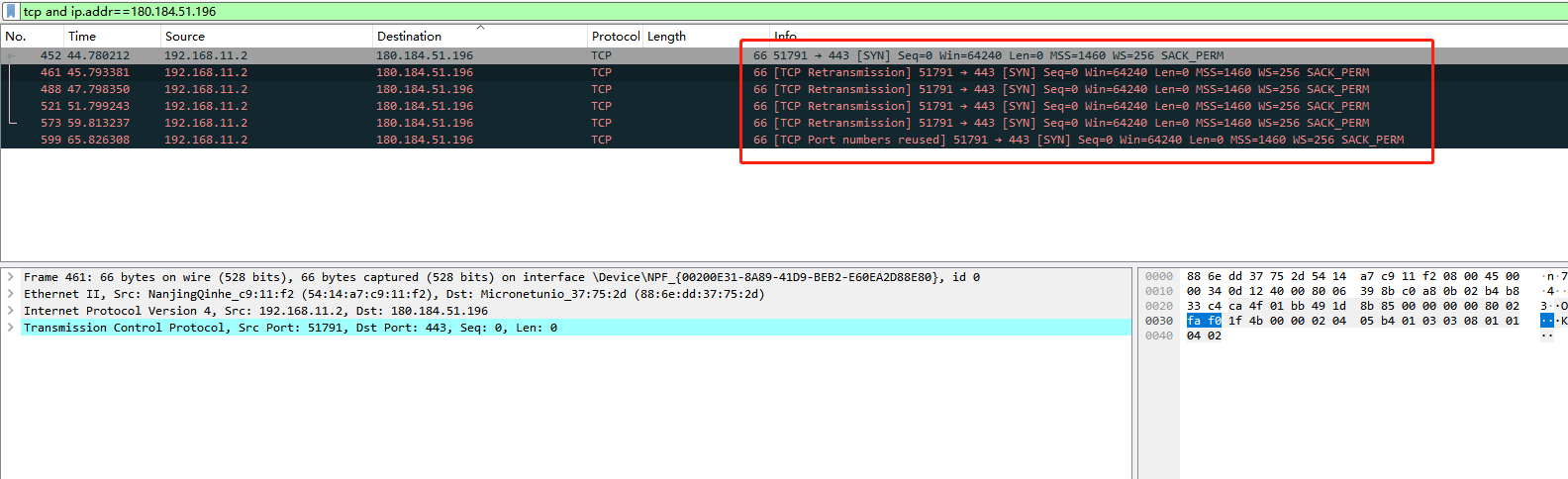
### 1.7 TCP连接异常

1. **建立连接时，客户端发给服务端的SYN请求报文丢失**

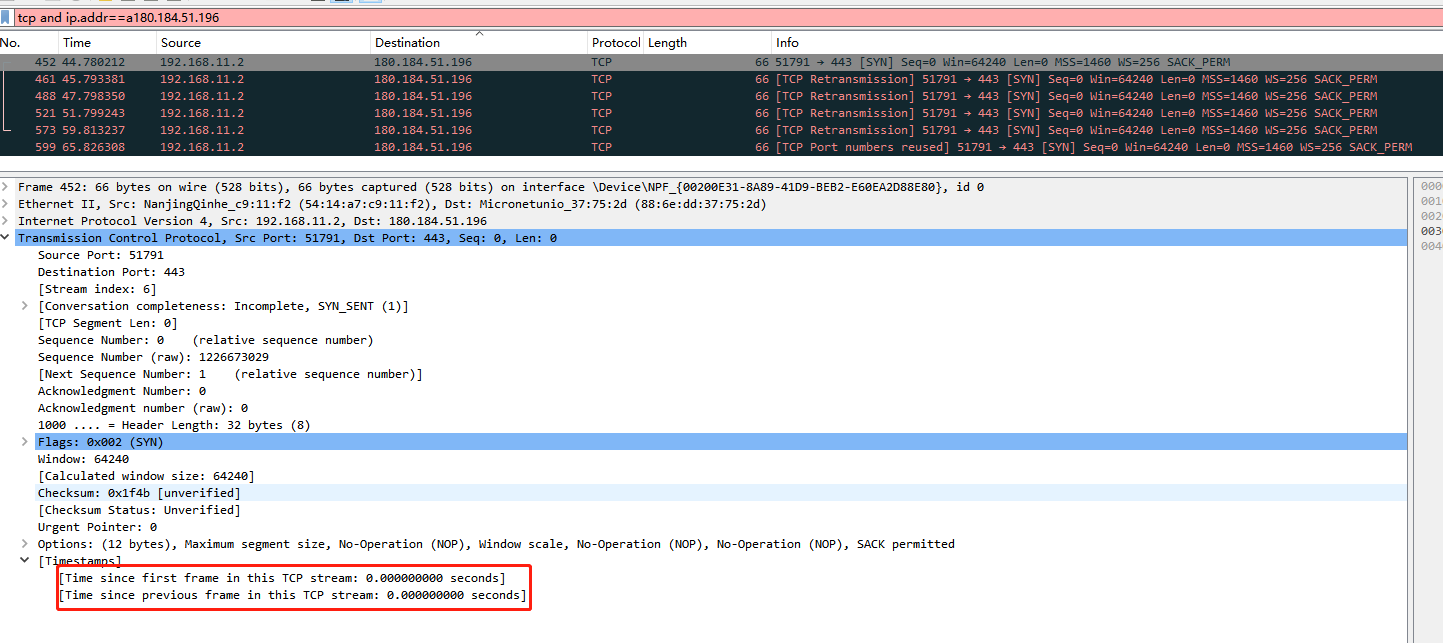
客户端发出SYN报文后，就会在本地启动后一个超时重传计时器，若超过所设定的时长还未收到服务端的回应（可能是ACK丢失；也可能是SYN丢失；也可能是服务端设置防火墙，直接给收到的SYN包丢了），则客户端就一直超时重传 SYN 报文，直到达到最大的重传次数，达到最大重传次数后对端若仍无响应则直接进入CLOSED状态

重传机制：按指数退避计算重传周期，如下的抓包信息

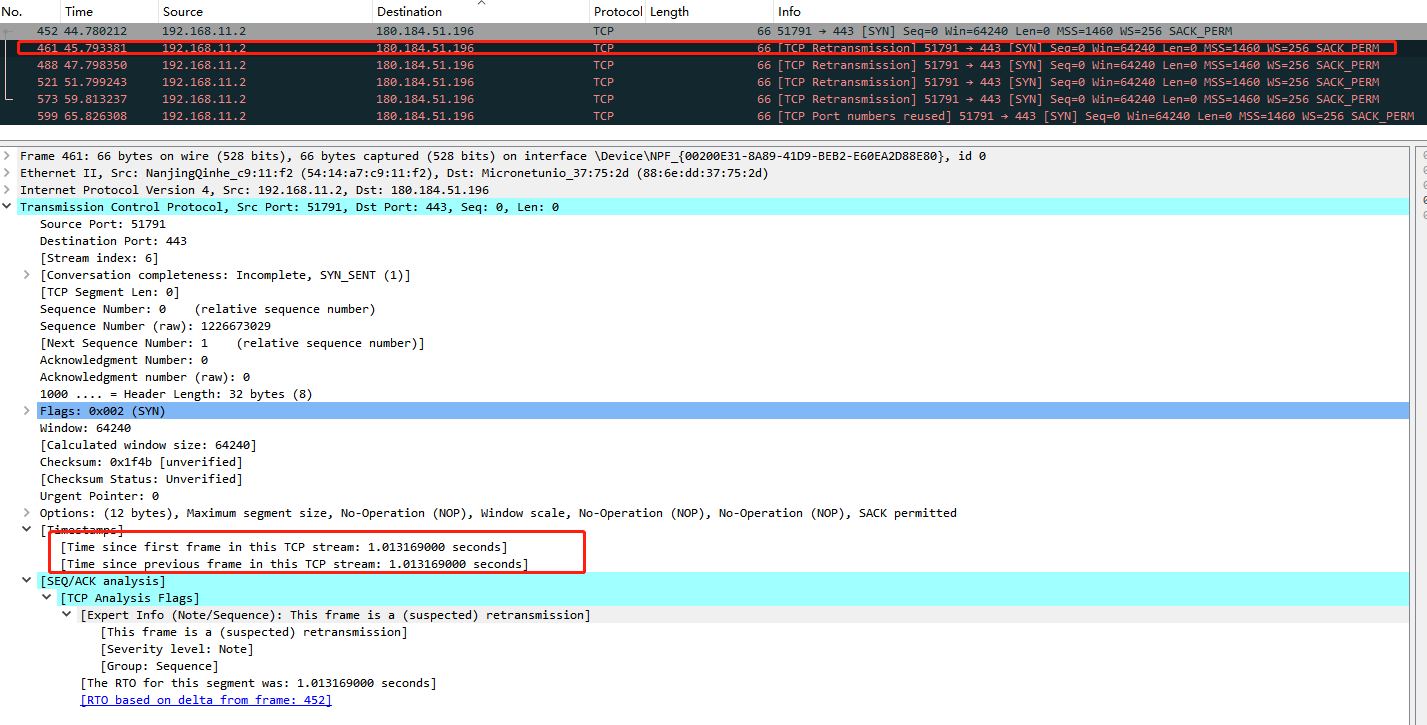
最大重传次数：由内核决定



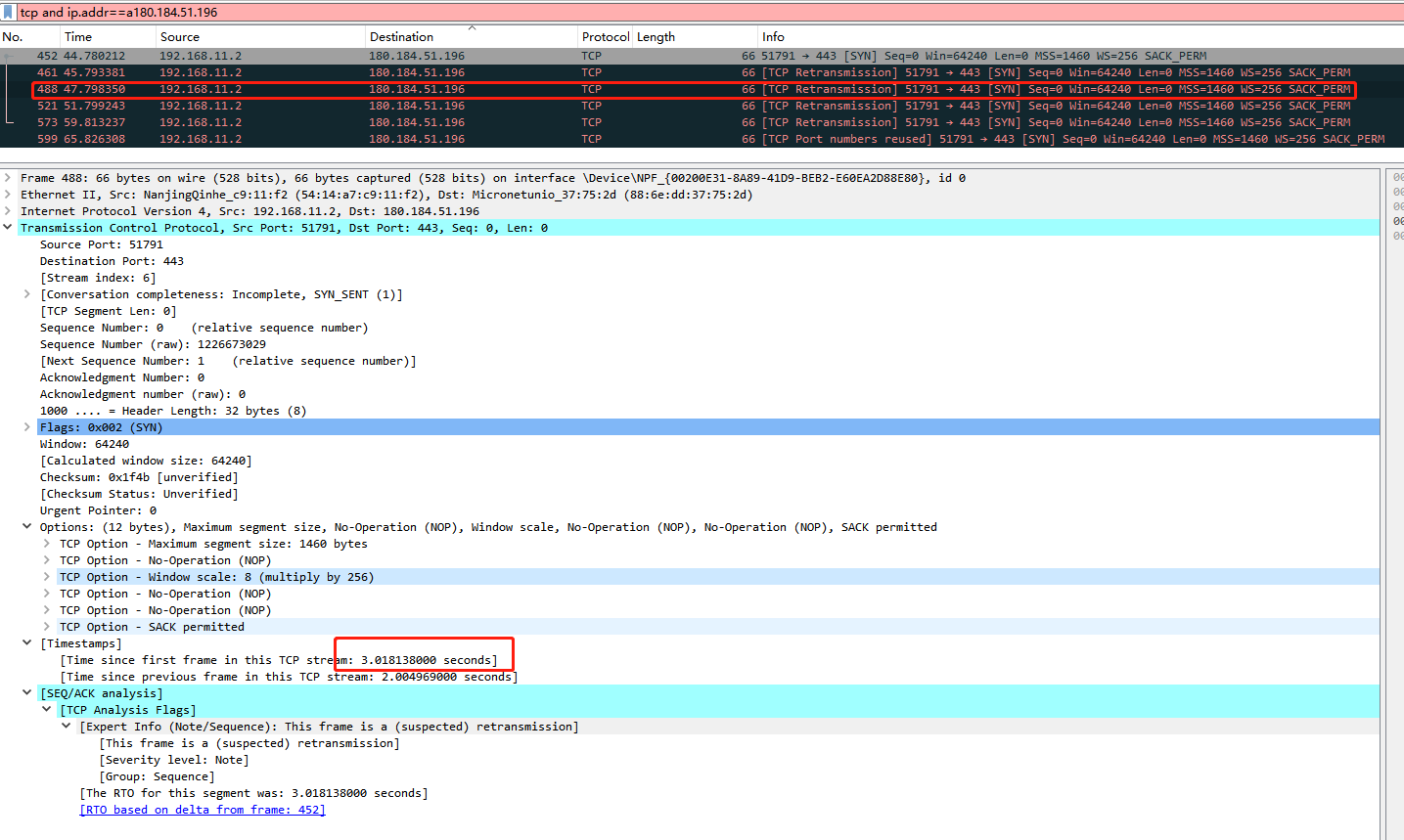
第一个SYN报文



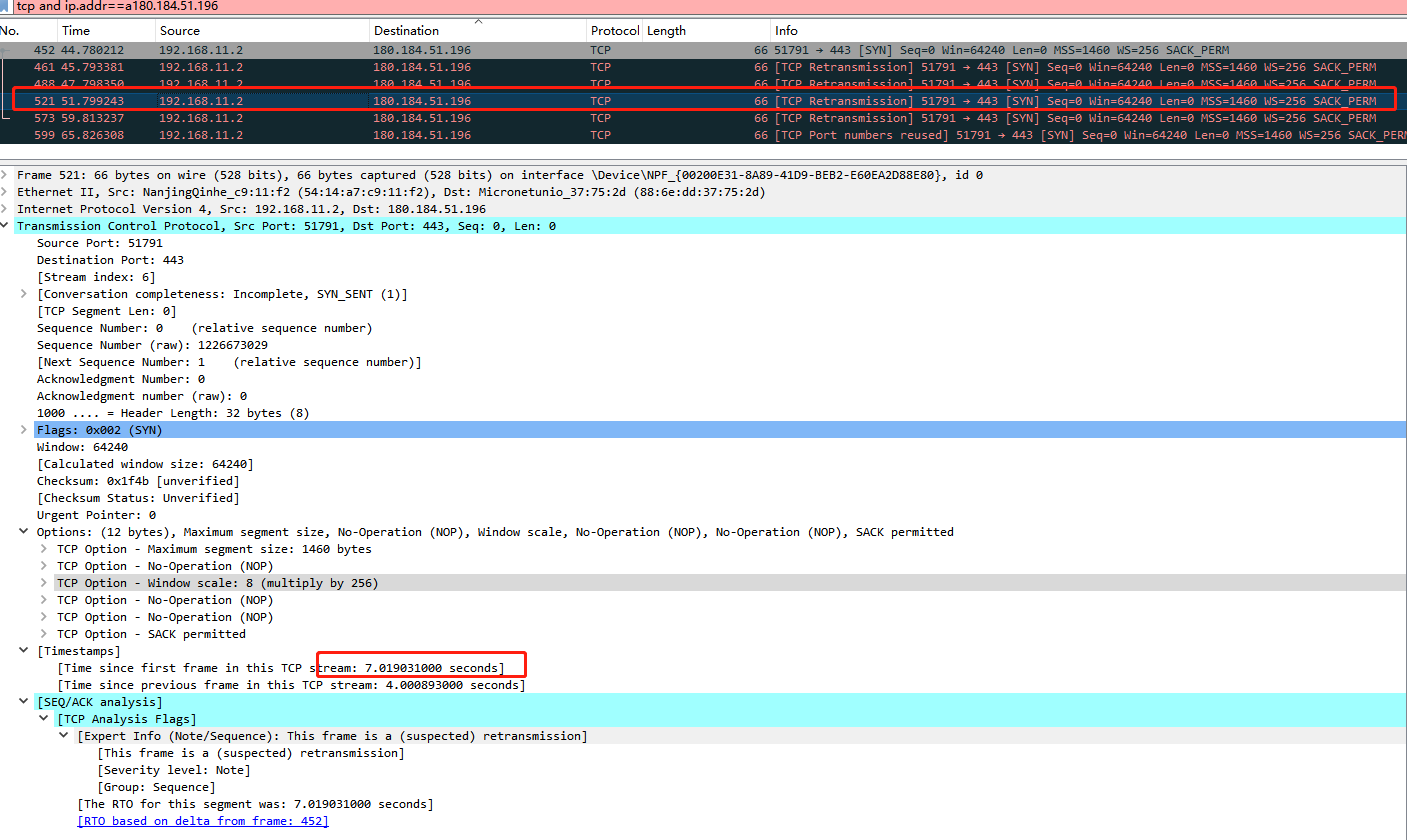
第二个SYN重传报文：间隔第一个SYN报文1秒



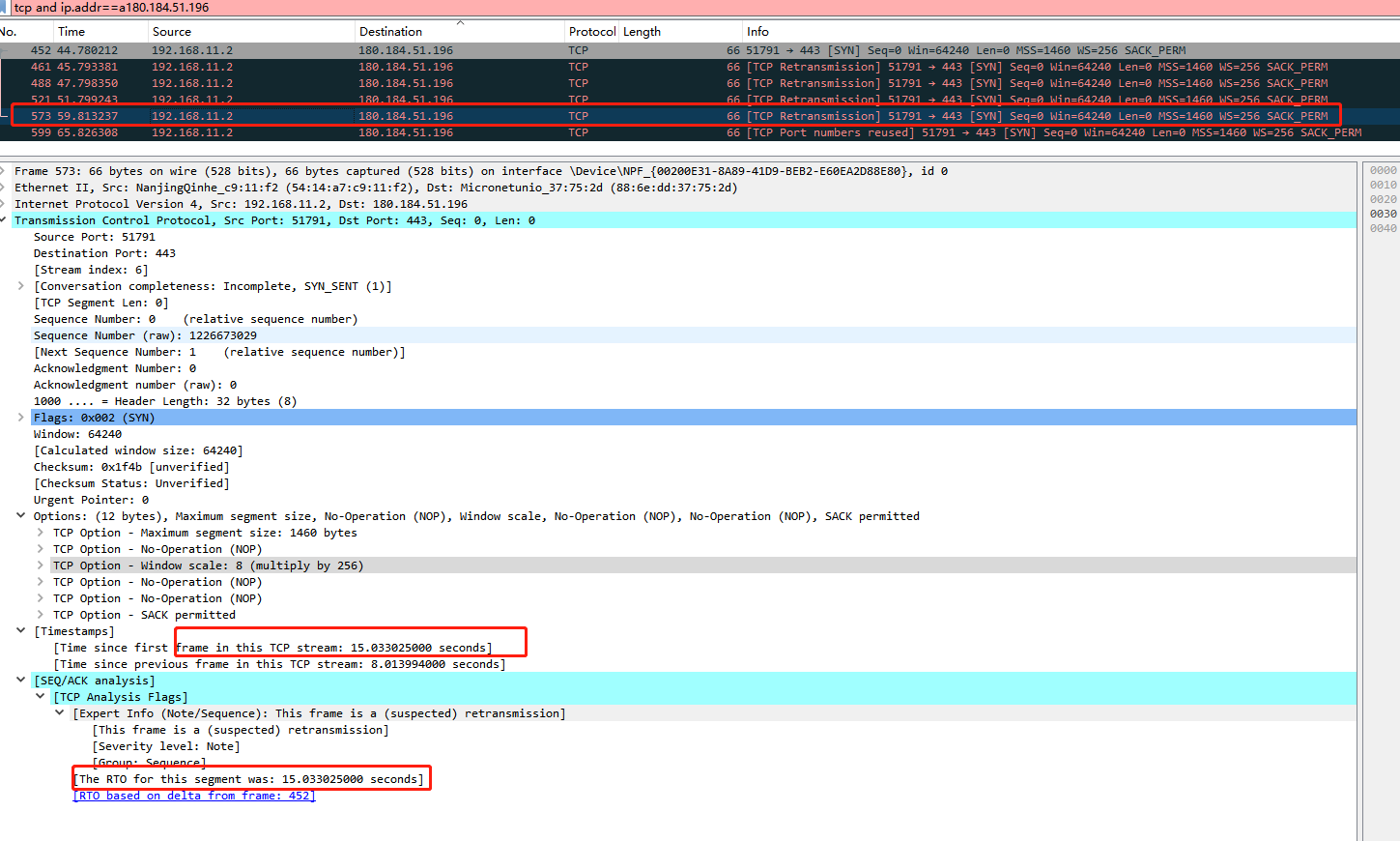
第三个SYN重传报文：间隔第一个SYN报文3秒



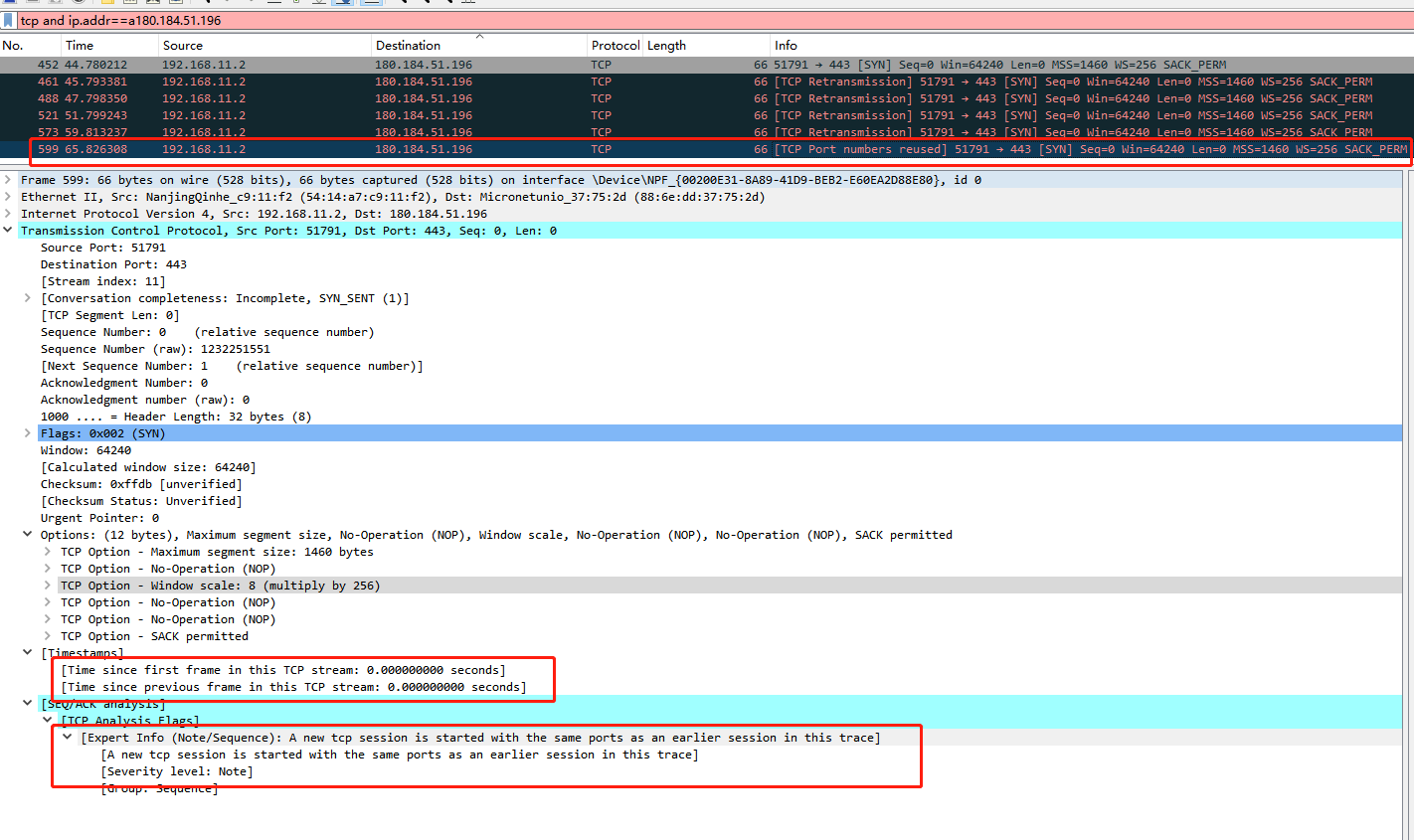
第四个SYN重传报文：间隔第一个SYN报文7秒



第五个SYN重传报文：间隔第一个SYN报文15秒



重启会话



什么情况会导致SYN请求报文的丢失呢？

(1)开启 tcp\_tw\_recycle 参数，并且在 NAT 环境下，造成 SYN 报文被丢弃

(2)accpet 队列满了，造成 SYN 报文被丢弃

(3)SYN 队列满了，造成 SYN 报文被丢弃

1. **建立连接时，服务端发给客户端的确认ack报文丢失**

服务端在收到客户端的第一次握手SYN报文段后，将TCP状态置为 SYN\_RECV状态，并发送SYN+ACK报文段给客户端。

如果第二次握手的报文段丢失，服务端会发起重传；客户端由于收不到SYN+ACK，无法判断是第一次握手的SYN报文段丢失，还是第二次握手的SYN+ACK报文段丢失，所以客户端也会发起重传。

客户端的重传过程如上节所述；服务端的 SYN+ACK重传次数由参数 tcp\_synack\_retries 配置。

1. **建立连接时，客户端发给服务端的确认ack报文丢失**

客户端在收到服务端的第二次握手SYN+ACK报文段后，将TCP状态置为 ESTABLISH 状态，并发送ACK报文段给服务端。

如果第三次握手的ACK报文段丢失，则服务会重传SYN+ACK报文段，直到收到ACK响应或者达到最大重传次数。

注意： 第三次握手的ACK报文段没有重传，当ACK丢失，只能依靠服务端重传SYN+ACK报文段。

1. **第一次挥手报文丢失**

当客户端（主动关闭方）调用close()函数后，就会发送FIN报文给服务端（确切的说，调用close将会时fd的引用计数减1，当fd的引用计数为0时，触发TCP四次挥手关闭TCP连接），随后客户端将TCP状态置为 FIN\_WAIT\_1 的状态。

如果第一次挥手的FIN报文段丢失，则客户端会开启重传流程，重传次数由内核参数 tcp\_orphan\_retries 控制。

当第一次挥手的FIN报文段达到最大重传次数后，客户端将停止重传，直接进入 CLOSE 状态。

1. **第二次挥手报文丢失**

当服务端收到第一次挥手的FIN报文段后，将本端TCP状态置为 CLOSE\_WAIT，并向客户端回复第二次挥手的ACK报文段。

上面已经提到，ACK报文段是不会重传的，所以如果ACK丢失，客户端会重传FIN报文，直到收到ACK报文，或达到FIN的最大重传次数。

另外还有一种情况需要注意：

如果客户端成功接收到第二次挥手的ACK报文，则客户端的TCP状态置为 FIN\_WAIT\_2 ，此时客户端开始等待服务端的第三次挥手FIN报文。

FIN\_WAIT\_2状态不会持续太久，内核参数 tcp\_fin\_timeout 控制其时长，默认是 60秒。

对于调用close()关闭的连接，如果在 60秒 后还没有收到第三次挥手的FIN报文，客户端连接会直接进入 CLOSE 状态。

1. **第三次挥手报文丢失**

当服务端收到客户端的第一次挥手的FIN报文后，内核会自动的回复ACK，并置TCP状态为 CLOSE\_WAIT，顾名思义，CLOSE\_WAIT状态表示内核在等待应用进程调用clsoe()函数关闭连接（wait to close）。

此时，内核没有权利替代进程关闭连接，必须由进程主动调用close函数来触发服务端发送FIN报文。

当服务端调用close()函数后，服务端发出第三次挥手报文，TCP状态将从 CLOSE\_WAIT 状态转入 LAST\_ACK 状态，如果在超时时间内未收到客户端的第四次挥手 ACK报文，则重传FIN报文，最大重传次数由内核参数 tcp\_orphan\_retries 控制。

CLOSE\_WAIT状态的最大持续时间是多少：

CLOSE\_WAIT状态表示的就是内核在等待应用进程调用close()函数来关闭连接，至于应用进程什么时候close，这完全取决于程序。所以，从理论上来讲，只要被动关闭端不断电，进程不退出，那么CLOSE\_WAIT状态就会一直持续下去，CLOSE\_WAIT的最大持续时间从理论上来讲可以达到无限长。

1. **第四次挥手报文丢失**

当客户端收到服务端的第三次挥手FIN报文后，TCP状态由 FIN\_WAIT\_2 转入 TIME\_WAIT，并回复第四次挥手ACK报文给服务端。

如果第四次挥手ACK报文丢失，ACK报文不会重传，服务端在超时未收到ACK后会重传FIN，直至成功收到ACK会达到最大重传次数。

TIME\_WAIT的最大持续时间默认为 60秒（2\*MSL），超时后TCP状态转为 CLOSED。

**几种TCP连接中出现RST的情况**

RST一般用来关闭异常的一条连接。通过在TCP头中将 RST标志位置为1，在接收端收到后就会关闭连接，此时一般会看到 “connection reset” 或 “connection refused” 的报错

**1、端口未打开**

服务器程序端口未打开而客户端来连接。在某些情况下，操作系统也会完全不理会这些发到未打开端口请求。

比如主机241向主机114发送一个SYN请求，表示想要连接主机114的40000端口，但是主机114上根本没有打开40000这个端口，于是就向主机241发送了一个RST。这种情况很常见。特别是服务器程序core dump之后重启之前连续出现RST的情况会经常发生

1. **请求超时**

在TCP连接建立或数据传输过程中，如果一方长时间没有收到对方的响应（如SYN-ACK、ACK等），可能会认为连接已经失效，从而发送RST报文来终止连接。例如，在三次握手过程中，如果客户端发送了SYN报文后，服务器由于某种原因（如端口未开放、资源不足等）无法响应SYN-ACK，客户端可能会超时并发送RST报文，请求连接方发起

1. **提前关闭**

TCP协议栈在检测到任何异常时，可能会发送RST报文来终止连接。这些异常可能包括序列号错误、ACK确认错误等，导致TCP协议栈无法继续维持连接。

1. **[在一个已关闭的socket上收到数据](http://my.oschina.net/costaxu/blog/127394" \l "OSC_h2_4" \t "https://blog.csdn.net/yang_chen_shi_wo/article/details/_blank)（半打开连接）**

半打开连接是指已经建立的连接中，一端已经关闭（如发送了FIN报文并收到ACK），但另一端仍然认为连接是打开的。在这种情况下，如果仍然尝试向已经关闭的一端发送数据，可能会收到RST报文作为响应

1. **用于拒绝一个非法连接**

当某些恶意程序或攻击者尝试建立非法的TCP连接时，被连接的一方可能会发送RST报文来阻止该连接。这有助于保护系统免受潜在的网络攻击。

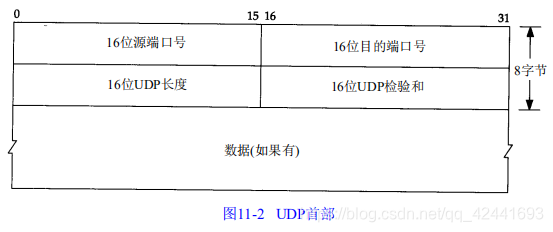
## UDP（User Datagram Protocol）协议

### 2.1 UDP概述

UDP（User Datagram Protocol，用户数据报协议） 是一种无连接的传输层协议，提供面向事务的简单不可靠信息传送服务。它常用于一次性传输较少数据的网络应用，如DNS、SNMP等。UDP不需要建立连接，不会引入建立连接的时延，但不保证向上层协议提供消息传递。UDP的首部格式包括UDP首部和用户数据，而UDP校验和检查首部和数据部分。

尽最大努力交付，没有拥塞控制，支持一对一、一对多、多对一和多对多的交互通信。

### 2.2 UDP的首部格式



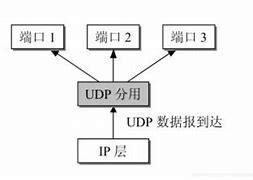
源端口号：在需要对方回信时选用，不需要时可全用0

目的端口号：在终点交付时必须使用

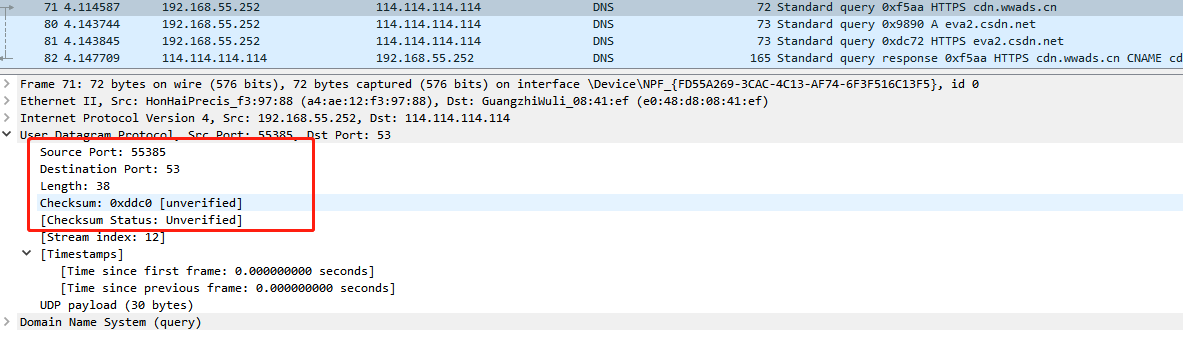
长度：UDP用户数据报的长度，最小值是8

检验和：检测UDP用户数据报在传输中是否有错，有错就丢弃

UDP基于端口的分用



### 2.3 UDP的报文抓包验证



## 无线通信

### 1、什么是无线通信

## 四、WIFI

### 1、什么是WiFi，它代表的是什么？

WiFi是一种基于IEEE802.11系列协议标准实现的无线通信技术，该通信协议于1996年由澳洲的研究机构CSIRO提出，WiFi 凭借其独特的技术优势，被公认为是目前最为主流的WLAN技术标准。随着WiFi无线通信技术的不断优化和发展，当前主要有4种通信协议标准，即802.11g、802.11b、802.11n和802.11a，根据不同的协议标准主要有两个工作频段，分别为2.4GHz和5.0GHz。

WiFi主要研究的是OSI网络模型的最下面两层：数据链路层(Data link layer)和物理层(Physical layer)

### 2、WLAN与WIFI

WLAN（Wireless Local Area Network，无线局域网），WLAN是指利用射频(Radio Frequency , RF)技术，使用电磁波并利用无线信道作为传输介质的计算机网络。以此所构成的局域网络，在空中进行通信连接。WLAN技术的出现是为了弥补有线网络的不足，以达到网络延伸到目的，实现无网线、无距离限制的通常网络。目前常用的WLAN实现技术有蓝牙、zigbee、WiFi等等。需要注意的是，目前使用较广的是WLAN技术是WIFI，但是WiFi并不等同于WLAN，它只是WLAN实现技术的一种。

WiFi（Wireless Fidelity，无线保真），WiFi是一个国际无线局域网（WLAN）标准，全称为Wireless Fidelity。WiFi基于IEEE802.11协议族，IEEE802.11协议最初发表于1997年，定义了WLAN的MAC层和物理层标准。继802.11协议发表后，又相继推出了802.11a、8021.11b、802.11g、802.11n等。WiFi是一个无线网络通信技术的品牌，由WIFI联盟所持有，使用在符合IEEE802.11标准上，目标是改善基于IEEE802.11标准的无线产品之间的互通性。

所谓互通性也就是说，不同芯片厂商，产品商的WiFi产品可以基于IEEE802.11标准之下无障碍，流畅的进行通信交互。为了达到这个目的，WiFi联盟（WiFi Alliance，简称WFA）提出市面上流通的WiFi产品需要去他们指定的认证实验室进行认证测试。当然，这些都是收费的。那么是不是也可以不用去？也可以的，但是很多国家和地区会要求在本地售卖的WiFi产品过WFA认证。这也是为了更好的统一各个芯片厂商，使得大家生产出来的产品能够更好的互通互联，目前市场上的产品基本都有WIFI联盟的认证logo。

## 五、IPv6协议

## 六、蓝牙通信技术

## 七、IGMP(Internet Group Management Protocol)

### 7.1 IP组播基础

**组播IP** D类：224.0.0.0-239.255.255.255（1110）；虽然目前CIDR无类域间路由和VLSM可变长子网掩码的出现淡化了IP分类，但组播IP范围并未发生变化。

其中组播IP又可进行如下划分：

224.0.0.0-224.0.0.255：预留永久组播地址，通常为协议所使用。

224.0.1.0-231.255.255.255 和 233.0.0.0-238.255.255.255：ASM(Any Source Multicast)模型使用

232.0.0.0-232.255.255.255：SSM(Source-Specific Multicast)模型使用

239.0.0.0-239.255.255.255：本地管理组地址，也即私网组播IP。

常用协议组播地址：

|  |  |
| --- | --- |
| 组播地址 | 使用者 |
| 224.0.0.1 | 所有主机及路由器监听地址 |
| 224.0.0.2 | 所有路由器监听地址 |
| 224.0.0.4 | DVMRP协议使用 |
| 224.0.0.5  224.0.0.6 | OSPFv2协议使用 |
| 224.0.0.9 | RIPv2协议使用 |
| 224.0.0.12 | DHCP协议特定场景使用 |
| 224.0.0.13 | PIM协议使用 |
| 224.0.0.14 | RSVP协议特定场景使用 |
| 224.0.0.18 | VRRPv2协议使用 |
| 224.0.0.22 | IGMPv3协议使用 |
| 224.0.0.251 | mDNS协议使用 |
| 224.0.0.252 | Link-local Multicast Name Resolution协议使用 |

**组播MAC**

MAC地址为48bits，6字节。

1、第1个字节的最后1个bit=1，表明该MAC为组播MAC。否则表明为单播MAC。

2、MAC地址的前3个字节 24bits 称为 OUI (Organizationally unique identifier，厂商标识)，需要向IEEE购买分配。后 24bits，由厂家自行分配。

3、组播MAC由于没有明确的目的主机，因此规定 组播MAC由组播IP映射 而来。映射规则如下：

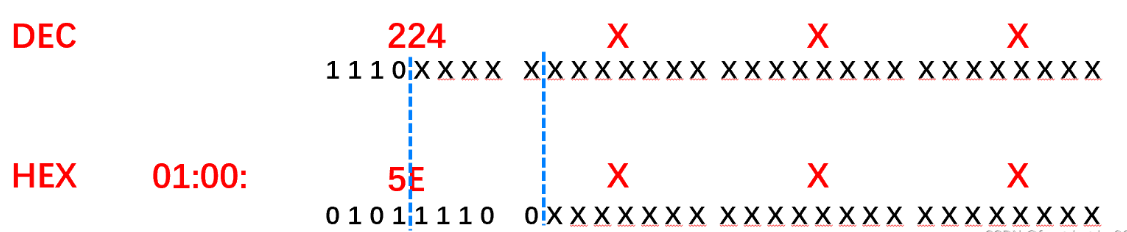
组播MAC的高 24bit = 0x01005e；

组播MAC的第 25位bit 固定为0；

组播MAC的剩余后 23bit 由组播IP的后 23bit 直接对应而来。

组播IP实际上有 32-4=28bits，因此存在 5bits 数据无法进行对应

因此实际上每2^5=32个组播IP对应一个组播MAC。



而且由于错配的bit位是高bit位的IP，所以通常对应同一个组播MAC的32个IP是不连续的。需要进行2进制转换核对。

例如：224.1.0.1、224.129.0.1、225.1.0.1、225.129.0.1、226.1.0.1、226.129.0.1、…、239.1.0.1、239.129.0.1具有相同的组播MAC。（后23bit完全一致）

注：对于IPv6的MIP和MMAC，情况稍有变化：IPv6组播MAC地址的高16位为0x3333，低32位为IPv6组播地址的低32位。常用的IPv6组播地址为ff00::/8

### 7.2 IGMP概念

IGMP，互联网组管理协议，是TCP/IP协议族中负责IPv4组播成员管理的协议，用来在IP主机和与其直接相邻的组播路由器之间建立、维护组播组成员关系。通过在接收者主机和与其直连的组播路由器上配置IGMP，可以实现主机动态加入组播组和组播路由器对本地网络组成员信息的管理。IGMP协议是IP组播在末端网络上使用的信令机制，分为两个部分功能：主机侧和路由器侧。所有参与组播传输的接收者主机必须应用IGMP协议。主机可以在任意时间、任意位置加入或退出组播组。组播路由器通过IGMP协议了解每个接口连接的网段上是否存在某个组播组的接收者，即组成员，而各主机只需要保存自己加入了哪些组播组。

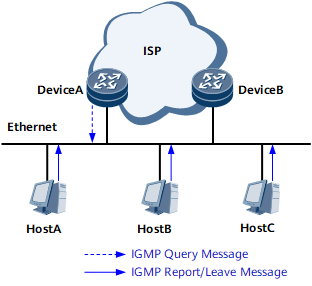
到目前为止，IGMP有三个版本，如下表所示：

|  |  |
| --- | --- |
| IGMP版本 | 支持的模型 |
| IGMPv1 | ASM（Any-Source Multicast）、需要通过SSM-Mapping技术来支持SSM（Source-Specific Multicast）。 |
| IGMPv2 | ASM、需要通过SSM-Mapping技术来支持SSM。 |
| IGMPv3 | ASM、SSM。 |

目的

要使组播报文最终能够到达接收者，需要将接收者主机接入IP组播网络，并加入到相应的组播组中。IGMP通过在主机侧和路由器侧交互IGMP报文实现组成员管理功能。IGMP协议可以记录接口下主机的加入和离开等信息，以确保组播数据能够正确地转发到该接口。

### 7.3 IGMP基本原理



IGMP报文大致分为三种，如上图所示：

**IGMP查询报文（IGMP Query）**：当路由器了解与该接口连接的网段上存在哪些组播组的接收者时，向主机发送的报文称之为IGMP Query报文。只有查询器（同一个网络中有多个多播路由器时，IGMP会选择其中一个来探寻主机间的成员关系，即查询器）负责发送IGMP Query报文。IGMP查询报文又细分为：

普遍组查询：不针对指定的组播源和组播组。

特定组查询：针对指定的组播组。

特定源组查询：针对指定的组播源和组播组。

**IGMP加入报文（IGMP Report）**：当主机想要加入组播组时，向上游发送的报文称之为IGMP Report报文。

**IGMP离开报文（IGMP Leave）**：当主机想要离开组播组时，向上游发送的报文称之为IGMP Leave报文。

注：IGMPv2和IGMPv3支持IGMP离开报文，IGMPv1不支持

使能了IGMP协议的组播设备在网段中的角色有两个：

**查询器**

负责发送查询报文，并接收主机反馈的加入报文和离开报文，以此来了解与该接口连接的网段上有哪些组播组存在接收者（即组成员）。

**非查询器**

只接收主机反馈的加入报文，了解与该接口连接的网段上有哪些组播组存在接收者，并根据网段中查询器的动作确定当前网段中有哪些组播组成员离开。

通常情况下一个网段只有一个查询器，因此组播设备之间需要用某些方式来选出查询器。查询器选举时采用以下的原则：

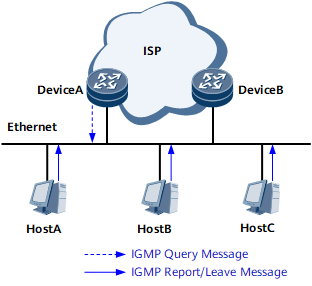
1. 组播设备A使能IGMP协议后，在IGMP协议启动阶段会默认自己为当前网段中的查询器，向网段中发送查询报文。如果收到IP地址比自己小的组播设备B发来的查询报文，则A由查询器转为非查询器，并启动其他查询器存在定时器，记录B为当前网段的查询器。
2. 如果组播设备A在非查询器状态时，收到查询器组播设备B发送的查询报文，则更新其他查询器存在定时器。如果此时收到的查询报文不是先前记录的查询器B发来的，而是新的组播设备C发来的，且C的IP地址比B的小，则更新查询器为C，同时更新其他查询器存在定时器。
3. 如果组播设备A在非查询器状态时，其他查询器存在定时器超时，则由非查询器转为查询器状态，承担起查询器的职责。

注：协议规定IGMPv1不支持查询器选举，IGMPv1查询器由上层协议（如PIM）指定。当前仅支持同网段上同版本的组播设备之间进行查询器选举。为了保证正常工作，需要在同网段所有组播设备上配置相同版本的IGMP。

**IGMP协议的实现过程**

通过发送查询报文并接收主机反馈的加入报文和离开报文，路由器可以了解与该接口连接的网段上存在哪些组播组的接收者（即组成员）。如果出现组成员，组播路由器应将对应组的组播数据报文转发到这个网段；如果没有组成员则不转发。主机可以自主决定加入或退出某个组播组。

如下图所示，使能了IGMP协议的路由器DeviceA会自动变为查询器并定时发出IGMP查询报文，与DeviceA在同一网段的所有主机（HostA、HostB、HostC）都能收到它发出的查询报文。



* 主机收到查询报文的处理：
* 如果主机之前已经加入了组G，就会在路由器允许的响应时间内随机发送一个组G的IGMP加入报文。DeviceA收到组G的IGMP加入报文会记录组G的相关信息，同时对该组启动一个定时器（如果已经启动则刷新定时器），以便长时间没有主机响应时切断该组的组播流量。DeviceA把组G的组播数据转发到HostA和DeviceA相连的接口所在的网段上。
* 如果主机不是任何组播组成员，主机在收到IGMP查询报文时不作任何响应。
* 主机加入组播组：
* 一个新加入组播组G的主机会主动发送一个该组的IGMP加入报文，通知路由器更新组播组信息，后续的加入报文则由路由器的查询报文来驱动。
* 主机离开组播组：
* 如果一个主机决定离开某个组播组G，会主动发送组G的IGMP离开报文。路由器收到后会触发一个指定组G的查询，确定该组在当前网段接收者的存在情况。如果在查询结束后仍然没有收到主机针对该组的IGMP加入报文，则删除已记录的组信息，停止转发该组数据到对应接口所在的网段上。

IGMP各版本处理IGMP报文的特点

|  |  |
| --- | --- |
| IGMP版本 | 特点 |
| IGMPv1 | * IGMPv1协议主要基于查询和响应机制完成组播组管理，支持查询和加入报文，处理过程与IGMPv2相同。主机离开组播组时不主动发送离开报文，待维护组成员关系的定时器超时后，路由器删除组记录。 * 只支持普遍组查询。 |
| IGMPv2 | * 运行IGMPv2的主机发送的IGMP报告中仅携带组信息，不能携带组播源信息。且一个报文中只能携带一个组记录。当主机发送一个组的IGMP加入报文给路由器后，路由器会通知组播转发模块，以便这个组的组播数据到来时能够正确转发给该主机。 * IGMPv2协议具有Report报文抑制机制，可以减少网络中的IGMP的重复Report报告。例如： * 当一个主机HostA加入了某个组播组G，在收到路由器的查询报文后，HostA会在0～最大响应时间（查询报文中已经指定）之间选择一个随机值作为定时器的超时时间，并在该定时器超时后，向路由器发送组G的加入报文。如果在超时时间内，HostA收到了加入同一个组的主机HostB发送的加入报文，则HostA不再向路由器发送组G的加入报文。 * 当主机退出某个组G时，会向路由器发送一个指定组G的IGMP离开报文。由于IGMPv2报告抑制机制，路由器无法确定是否还有其他主机加入了组G。这时路由器会触发一个指定组G的查询，如果其他主机加入了组G，就会发送针对组G的IGMP加入报文。如果路由器发送了若干次数指定组G的查询之后，仍然没有收到主机针对组G的IGMP加入报文，那么路由器就不再记录组G的信息，停止转发该组数据到对应接口所在的网段。 * 除了支持普遍组查询，还支持特定组查询。   说明：  IGMP的查询器和非查询器都会处理IGMP组加入信息，但是只有查询器负责发送查询报文。IGMP非查询器不处理IGMPv2离开报文。 |
| IGMPv3 | * IGMPv2报文中只能携带组播组的信息，不能携带组播源的信息。这样运行IGMPv2的主机就只能选择加入某个组，而不能选择加入某个组播源/组。IGMPv3解决了该问题。运行IGMPv3的主机不仅能够选择组，还能根据需要选择组播源/组。主机发送的IGMPv3报文中可以包含多个组记录，每个组记录中可以包含多个组播源。在路由器侧，查询器发送查询报文并接收主机反馈的加入报文和离开报文，以此来了解与该接口连接的网段上有哪些组播组存在接收者，并将组播数据转发到相应的网段。IGMPv3的组记录有include和exclude二种组过滤模式。 * 在include模式下   处于激活状态的组播源表示需要路由器转发这个源的数据。  不活动的源会被路由器删除并停止转发这个源的数据。   * 在exclude模式下   处于激活状态的组播源表示处于冲突域中。也就是说，与该路由器接口同一网段的主机中，有的主机需要该源的数据，有的主机不需要该源的数据，在这种情况下该源的数据仍然需要转发。  不活动的组播源表示不需要转发该源的数据。  组中没有记录的组播源的数据全部都要转发。   * IGMPv3没有Report报文抑制机制，所有加入组播组的主机在收到查询时都会发送IGMP加入报文作为响应。 * 由于有了对组播源的选择，IGMPv3设备在普遍组查询和特定组查询的基础上增加了特定源组查询，用以在收到特定组播源的数据时确定是否存在该数据的接收者。 |
| IGMPv2较IGMPv1的优势 | * 报文类型包括成员离开报文，可以更及时有效管理组播组成员。 * 可以直接对特定的组播组进行选择，增加了选择的粒度。 |
| IGMPv3较IGMPv2的优势 | * 可以直接对特定的组播源进行选择，增加了选择的粒度。 * 一个报文中可以携带多个组记录，减少了网段中的IGMP报文数量。 |

**IGMP策略控制**

IGMP策略控制是对IGMP的行为进行限制或者扩充的附加功能，对IGMP协议本身实现没有影响。IGMP策略控制包括IGMP-Limit、IGMP报文源地址过滤和Group-Policy。

IGMP-Limit

通过在与用户相连的路由器接口上配置IGMP表项限制，限制组播组或源组的数量，使加入组播组的用户收看更加清晰稳定的节目。

IGMP报文源地址过滤

通过IGMP报文源地址过滤特性可以防止伪造的IGMP报文的非法攻击，确保组播网络的安全性。

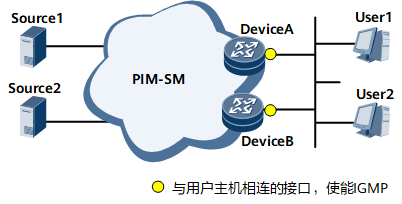
Group-Policy

Group-Policy是管理员配置在路由器接口上的一种过滤组策略。配置Group-Policy之后，路由器可以对某些特定的组进行限制，不建立对应的表项，从而提高IGMP的安全性。

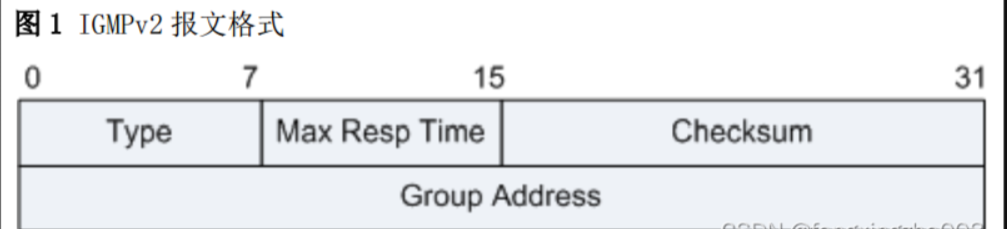
IGMP典型应用

IGMP是处理主机加入路由网络的协议。因此，该协议应用于路由边界与主机相连的区域。该协议可以处理多主机和多个组播设备分别使用不同版本时的情况。

下图是IGMP应用典型组网图



### 7.4 IGMP报文格式及相关命令



**Type**：对v1的Version和Type进行合并为Type字段，共8bits。

0x11时表示Membership Query IGMP 查询消息，通用组查询或特定组查询；

0x12时表示 Version 1 Membership Report IGMPv1 成员报告消息；

0x16时表示Version 2 Membership Report IGMPv2 成员报告消息；

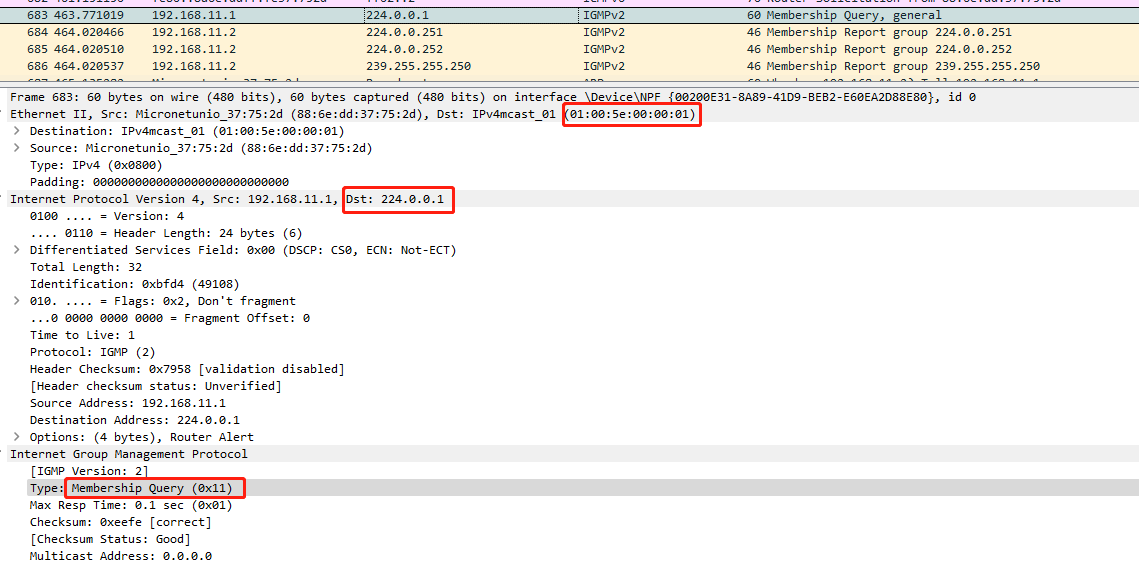
0x17时表示Leave Group 离组报文消息。

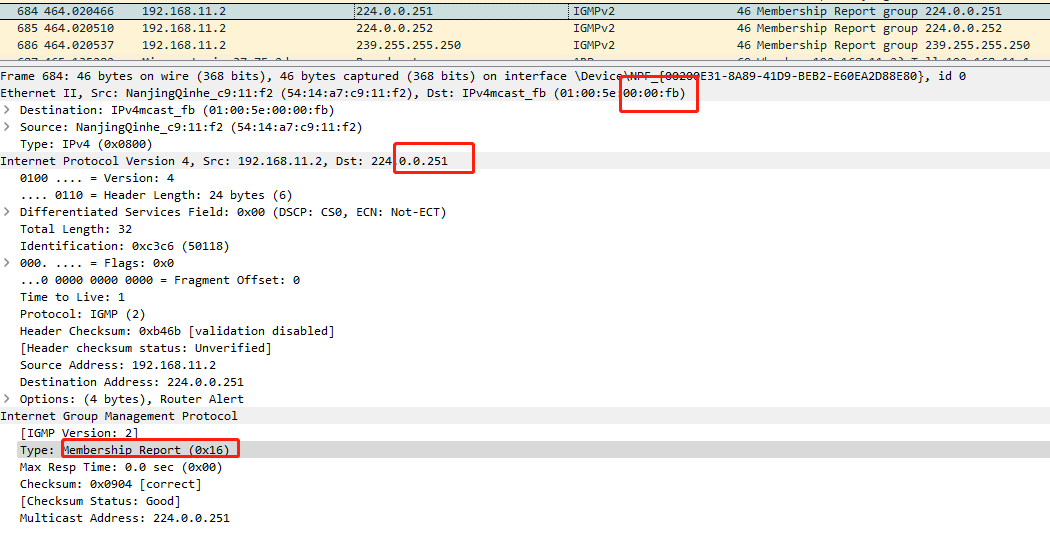
**Max Resp Time：**表示主机进行响应时的最大时间。主机对通用组报文查询时随机取值为(0，Max Resp Time]

**CheckSum：**用于对IGMP报文进行校验。

**Multicast Add：**组播组地址，在不同报文场景下具有不同含义

抓包验证



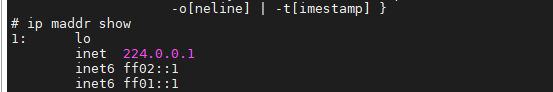


ip maddr :用来添加，改变，删除以及显示多播地址。



展示多播地址：







添加和删除多播地址：

固定的命令格式，必须指定device的名字ip maddr add mutil-addr dev device-name

注意，能手动加入的都是静态的链接层MAC组播地址，像IPv4和IPv6的组播地址只能动态加入，因此无法设置。加入的组播MAC地址，后面都带有static的标志。

同理，也只能删除静态的连接层MAC组播地址

### 7.5组播模型分类

根据接收者对组播源处理方式的不同，组播模型分为以下三类：

(1) **ASM 模型：Any-Source Multicast，任意信源组播**

在ASM 模型中，任意一个发送者都可以作为组播源向某组播组地址发送信息。众多接收者通过加入由该组播组地址标识的组播组以获得发往该组播组的组播信息。

在ASM 模型中，接收者无法预先知道组播源的位置，但可以在任意时间加入或离开该组播组。

(2) **SFM 模型：Source-Filtered Multicast，信源过滤组播**

该模型继承了ASM 模型，从发送者角度来看，两者的组播组成员关系完全相同。SFM 模型在功能上对ASM 模型进行了扩展。

在SFM 模型中，上层软件对收到的组播报文的源地址进行检查，允许或禁止来自某些组播源的报文通过。

因此，接收者只能收到来自部分组播源的组播数据。从接收者的角度来看，只有部分组播源是有效的，组播源被经过了筛选。

(3) **SSM 模型：Source-Specific Multicast，指定信源组播**

在现实生活中，用户可能只对某些组播源发送的组播信息感兴趣，而不愿接收其它源发送的信息。该模型为用户提供了一种能够在客户端指定组播源的传输服务。

SSM 模型与ASM 模型的根本区别在于：SSM 模型中的接收者已经通过其它手段预先知道了组播源的具体位置。SSM 模型使用与ASM/SFM 模型不同的组播地址范围，直接在接收者与其指定的组播源之间建立专用的组播转发路径。

### 7.6路由器组播路由协议

连接在局域网上的多播路由器还必须和互联网上的其他多播路由器协同工作，以便把多播数据报用最小代价传送给所有的组成员，这就需要使用多播路由选择协议。多播路由选择协议，用来在多播路由器之间传播路由信息。

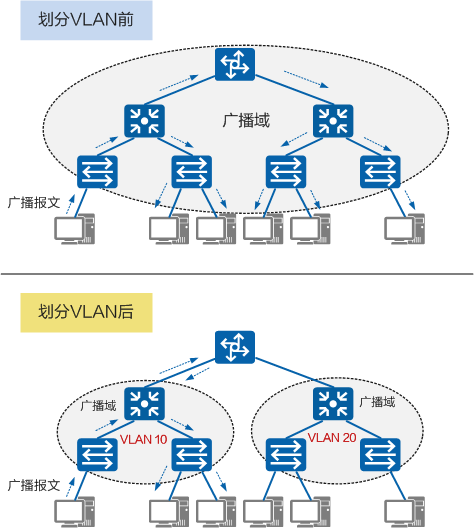
## VLAN

### 8.1 什么是VLAN

VLAN（Virtual Local Area Network）即虚拟局域网，是将一个物理的LAN在逻辑上划分成多个广播域的通信技术。每个VLAN是一个广播域，VLAN内的主机间可以直接通信，而VLAN间则不能直接互通。这样，广播报文就被限制在一个VLAN内。

### 8.2 为什么需要VLAN

早期以太网是一种基于CSMA/CD（Carrier Sense Multiple Access/Collision Detection）的共享通讯介质的数据网络通讯技术。当主机数目较多时会导致冲突严重、广播泛滥、性能显著下降甚至造成网络不可用等问题。通过二层设备实现LAN互连虽然可以解决冲突严重的问题，但仍然不能隔离广播报文和提升网络质量。在这种情况下出现了VLAN技术。这种技术可以把一个LAN划分成多个逻辑的VLAN，每个VLAN是一个广播域，VLAN内的主机间通信就和在一个LAN内一样，而VLAN间则不能直接互通，广播报文就被限制在一个VLAN内。如下图所示。



**VLAN作用**

* 限制广播域：广播域被限制在一个VLAN内，节省了带宽，提高了网络处理能力。

增强局域网的安全性：不同VLAN内的报文在传输时相互隔离，即一个VLAN内的用户不能和其它VLAN内的用户直接通信。

* 提高了网络的健壮性：故障被限制在一个VLAN内，本VLAN内的故障不会影响其他VLAN的正常工作。
* 灵活构建虚拟工作组：用VLAN可以划分不同的用户到不同的工作组，同一工作组的用户也不必局限于某一固定的物理范围，网络构建和维护更方便灵活。

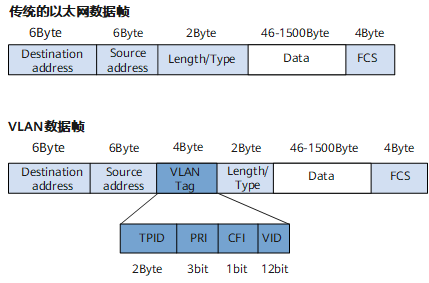
### 8.3 VLAN vs 子网

通过将IP地址的网络部分进一步划分为若干个子网，可以解决IP地址空间利用率低和两级IP地址不够灵活的问题。与VLAN相类似的是，子网也可以隔离主机间的通信。属于不同VLAN的主机之间不能直接通信，属于不同的子网的主机之间也不能直接通信。但二者没有必然的对应关系。



### 8.4 VLAN Tag和VLAN ID

要使交换机能够分辨不同VLAN的报文，需要在报文中添加标识VLAN信息的字段。IEEE 802.1Q协议规定，在以太网数据帧中加入4个字节的VLAN标签（又称VLAN Tag，简称Tag），用以标识VLAN信息。



EEE 802.1Q封装的VLAN数据帧格式

数据帧中的VID字段标识了该数据帧所属的VLAN，数据帧只能在其所属VLAN内进行传输。VID字段代表VLAN ID，VLAN ID取值范围是0～4095。由于0和4095为协议保留取值，所以VLAN ID的有效取值范围是1～4094。

交换机内部处理的数据帧都带有VLAN标签。而交换机连接的部分设备（如用户主机、服务器）只会收发不带VLAN tag的传统以太网数据帧。因此，要与这些设备交互，就需要交换机的接口能够识别传统以太网数据帧，并在收发时给帧添加、剥除VLAN标签。添加什么VLAN标签，由接口上的缺省VLAN（Port Default VLAN ID，PVID）决定。

### 8.5 VLAN的接口类型和VLAN标签的处理机制

现网中属于同一个VLAN的用户可能会被连接在不同的交换机上，且跨越交换机的VLAN可能不止一个，如果需要用户间的互通，就需要交换机间的接口能够同时识别和发送多个VLAN的数据帧。根据接口连接对象以及对收发数据帧处理的不同，当前有VLAN的多种接口类型，以适应不同的连接和组网。

不同厂商对VLAN接口类型的定义可能不同。对于华为设备来说，常见的VLAN接口类型有三种，包括：Access、Trunk和Hybrid。

Access接口

Access接口一般用于和不能识别Tag的用户终端（如用户主机、服务器）相连，或者不需要区分不同VLAN成员时使用。

在一个VLAN交换网络中，以太网数据帧主要有以下两种形式：

无标记帧（Untagged帧）：原始的、未加入4字节VLAN标签的帧。

有标记帧（Tagged帧）：加入了4字节VLAN标签的帧。

Access接口大部分情况只能收发Untagged帧，且只能为Untagged帧添加唯一VLAN的Tag。交换机内部只处理Tagged帧，所以Access接口需要给收到的数据帧添加VLAN Tag，也就必须配置缺省VLAN。配置缺省VLAN后，该Access接口也就加入了该VLAN。

当Access接口收到带有Tag的帧，并且帧中VID与PVID相同时，Access接口也能接收并处理该帧。

在发送带有Tag的帧前，Access接口会剥离Tag。

Trunk接口

Trunk接口一般用于连接交换机、路由器、AP以及可同时收发Tagged帧和Untagged帧的语音终端。它可以允许多个VLAN的帧带Tag通过，但只允许属于缺省VLAN的帧从该类接口上发出时不带Tag（即剥除Tag）。

Trunk接口上的缺省VLAN，有的厂商也将它定义为native VLAN。当Trunk接口收到Untagged帧时，会为Untagged帧打上Native VLAN对应的Tag。

Hybrid接口

Hybrid接口既可以用于连接不能识别Tag的用户终端（如用户主机、服务器）和网络设备（如Hub），也可以用于连接交换机、路由器以及可同时收发Tagged帧和Untagged帧的语音终端、AP。它可以允许多个VLAN的帧带Tag通过，且允许从该类接口发出的帧根据需要配置某些VLAN的帧带Tag（即不剥除Tag）、某些VLAN的帧不带Tag（即剥除Tag）。

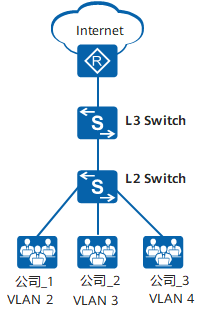
Hybrid接口和Trunk接口在很多应用场景下可以通用，但在某些应用场景下，必须使用Hybrid接口。比如在灵活QinQ中，服务提供商网络的多个VLAN的报文在进入用户网络前，需要剥离外层VLAN Tag，此时Trunk接口不能实现该功能，因为Trunk接口只能使该接口缺省VLAN的报文不带VLAN Tag通过。

### 8.6 VLAN的使用场景

VLAN的常见使用场景包括：VLAN间用户的二层隔离，VLAN间用户的三层互访。

VLAN间用户的二层隔离

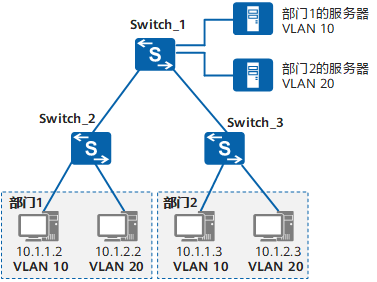
如下图所示，某商务楼内有多家公司，为了降低成本，多家公司共用网络资源，各公司分别连接到一台二层交换机的不同接口，并通过统一的出口访问Internet。



基于接口的VLAN划分组网图

为了保证各公司业务的独立和安全，可将每个公司所连接的接口划分到不同的VLAN，实现公司间业务数据的完全隔离。可以认为每个公司拥有独立的“虚拟路由器”，每个VLAN就是一个“虚拟工作组”。

再比如，某公司有两个部门，分别分配了固定的IP网段。为加强员工间的学习与交流，员工的位置有时会相互调动，但公司希望各部门员工访问的网络资源的权限不变。

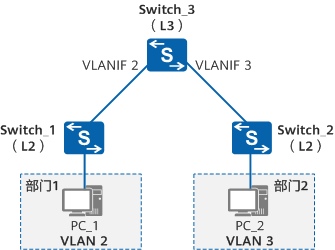


基于IP的VLAN划分组网图

为了保证部门内员工的位置调整后，访问网络资源的权限不变，可在公司的交换机Switch\_1上配置基于IP子网划分VLAN。这样，服务器的不同网段就划分到不同的VLAN，访问服务器不同应用服务的数据流就会隔离，提高了安全性。

VLAN间用户的三层互访

如下图所示，某小型公司的两个部门分别通过二层交换机接入到一台三层交换机Switch\_3，所属VLAN分别为VLAN2和VLAN3，部门1和部门2的用户互通时，需要经过三层交换机。



通过VLANIF实现VLAN间用户的三层互访

可在Switch\_1和Switch\_2上划分VLAN并将VLAN透传到Switch\_3上，然后在Switch\_3上为每个VLAN配置一个VLANIF接口，实现VLAN2和VLAN3间的路由。

VLAN的相关协议

IEEE 802.1Q

IEEE [802.1Q](https://info.support.huawei.com/info-finder/encyclopedia/zh/QinQ.html" \o "QinQ)（也被称为[Dot1q](https://info.support.huawei.com/info-finder/encyclopedia/zh/QinQ.html" \o "QinQ)）即Virtual Bridged Local Area Networks协议，规定了VLAN的实现标准。与标准的以太网数据帧相比，VLAN数据帧增加了1个4字节的VLAN标签。

LNP

链路类型协商协议（Link-type Negotiation Protocol，LNP）用来动态协商以太网接口的链路类型为Access或者Trunk。

以太网接口的链路类型协商为Access，缺省情况下加入VLAN1。

以太网接口的链路类型协商为Trunk，缺省情况下加入VLAN1～4094。

QinQ

[QinQ](https://info.support.huawei.com/info-finder/encyclopedia/zh/QinQ.html" \o "QinQ)（802.1Q-in-802.1Q）协议出自[IEEE 802.1ad](https://info.support.huawei.com/info-finder/encyclopedia/zh/QinQ.html" \o "QinQ)标准协议，通过在802.1Q标签报文的基础上再增加一层802.1Q的Tag来达到扩展VLAN空间的功能，可以使私网VLAN透传公网。

由于在骨干网中传递的报文有两层802.1Q Tag（一层公网Tag，一层私网Tag），即802.1Q-in-802.1Q，所以称之为QinQ协议。

云化场景下，VLAN存在的问题

随着网络技术的发展，云计算凭借其在系统利用率高、人力和管理成本低、灵活性和可扩展性强等方面表现出的优势，已经成为目前企业IT建设的新趋势。而服务器虚拟化作为云计算的核心技术之一，得到了越来越多的应用。

VLAN作为传统的网络隔离技术，在标准定义中VLAN的数量只有4096个，无法满足大型数据中心的租户间隔离需求。另外，VLAN的二层范围一般较小且固定，无法支持虚拟机大范围的动态迁移。

因此，RFC定义了VLAN扩展方案[VXLAN](https://info.support.huawei.com/info-finder/encyclopedia/zh/VXLAN.html" \o "VXLAN)（Virtual eXtensible Local Area Network，虚拟扩展局域网）。VXLAN采用MAC in UDP（User Datagram Protocol）封装方式，是NVO3（Network Virtualization over Layer 3）中的一种网络虚拟化技术。VXLAN完美地弥补了VLAN的上述不足，一方面通过VXLAN中的24比特VNI（VXLAN Network Identifier）字段，提供多达16M租户的标识能力，远大于VLAN的数量；另一方面，VXLAN本质上在两台交换机之间构建了一条穿越数据中心基础IP网络的虚拟隧道，将数据中心网络虚拟成一个巨型“二层交换机”，满足虚拟机大范围动态迁移的需求。