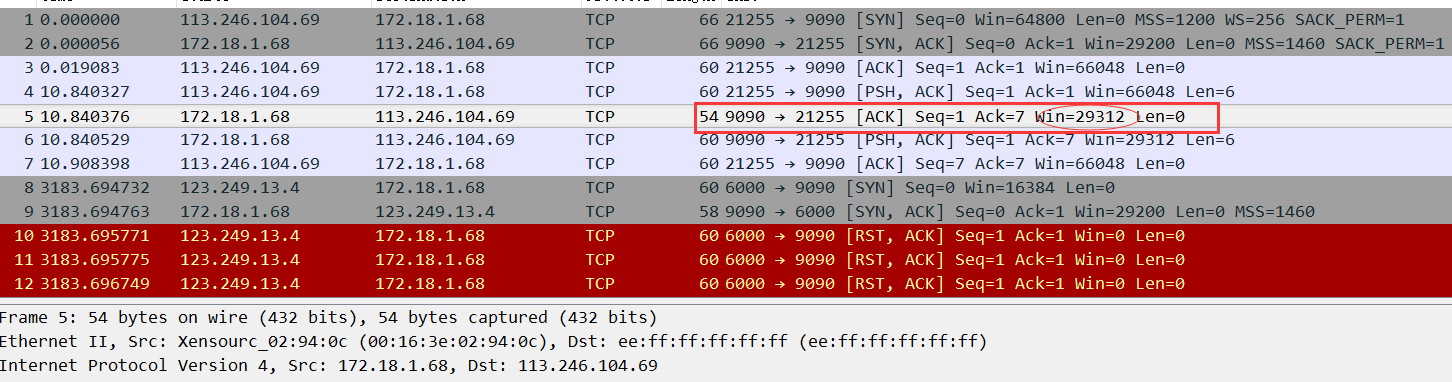
第18节课堂笔记

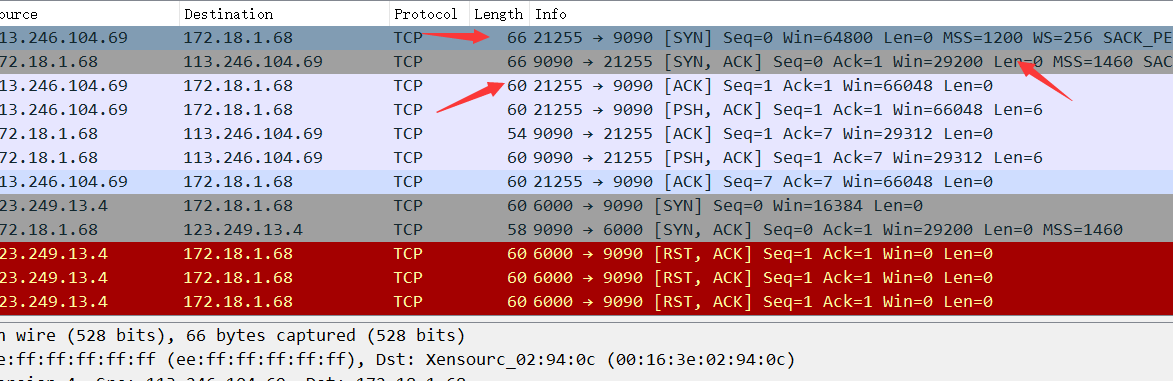
1. TCP滑动窗口：



接收端缓存的剩余空间，回复ACK时通告了发送端。



三次握手确认了接收端的窗口大小



如果连续有3次通告了发送端，窗口大小为0，那么，Zero window，不同的实现版本，表现的行为有点不一样，发送端会发送RST消息，去把这个TCP连接断掉。

TCPdump去转包。wirshar，脚本分析消息数据。lua shell python。

如果说，接收端频繁地通告发送一个较小的窗口，如果我们的发送端不停地去把这个窗口填满，那会导致网络上出现大量的小包。我们的带宽利用率时间上是很少的。（糊涂窗口）

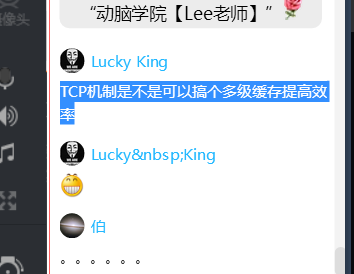
1. 是因为我们的接收引起的，一种情况就是去回复ack(0)，接收端收到这个ack的时候，就不会发送。或者说另外一种办法，就是等到合适的窗口大小，在通告发送端去发送一个较大的包。MSS（TCP包最大的大小， 1460个字节）。
2. 比如我说我们的回显程序，telnet，ssh，是有发送端引起的。Nagle算法，避免大量发送小包，发送小包的时候，等待40ms，是为了把大量的小包组装一个较大的包，然后发送一个较大的包。

TCP选项，TCP\_NODELAY。

zeromq，消息队列，分布式集群后台的一个异步通知的。发送缓存，一次性把发送缓存的数据包取出来，直到一个MSS的大小，或者取完了，然后组包发送。

后台性能优化方法之一： 批量操作 redis

控制发送的量，控制发送端的速率以适配接收端的处理消息的速率。

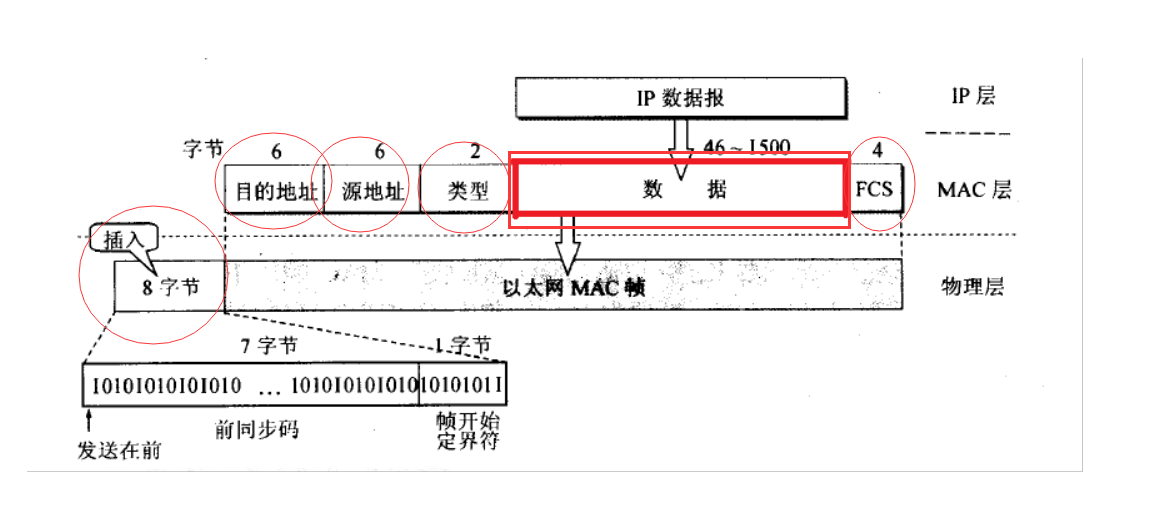


多级缓存就能提到效率呢？多次copy，每一次copy的代价你知道是多少？ zero copy？

# MTU

MTU（Max Transmit Unit）数据链层定义的。

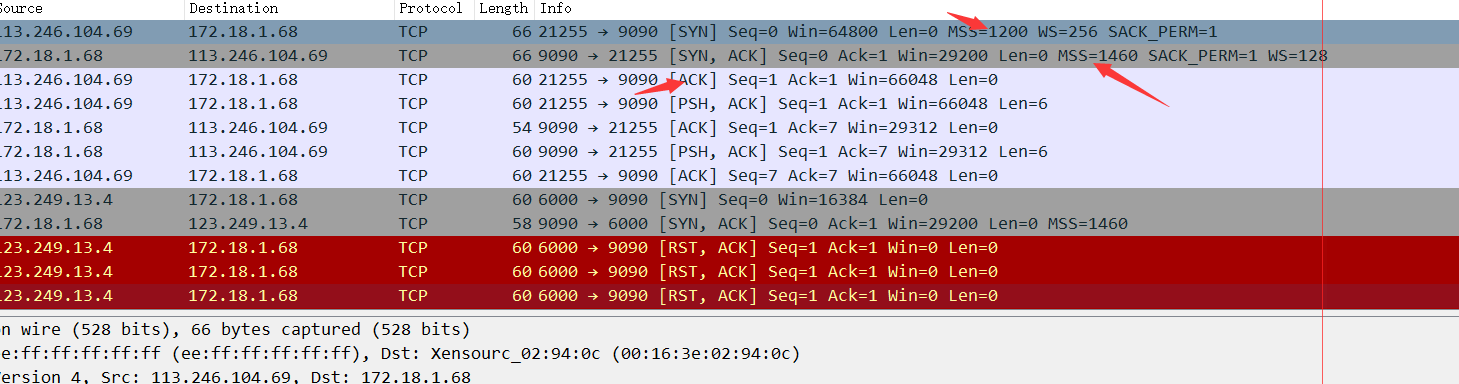
MTU = 1500个字节。TCP协议



MSS ： 最大传输分节（Max Segment Size），1460，MTU -ip头 - tcp头 = 1460。

建立TCP连接的时候，是由双方去协定的。服务器有一个SMSS，客户端CMSS

MSS = MIN(SMSS, CMSS)。

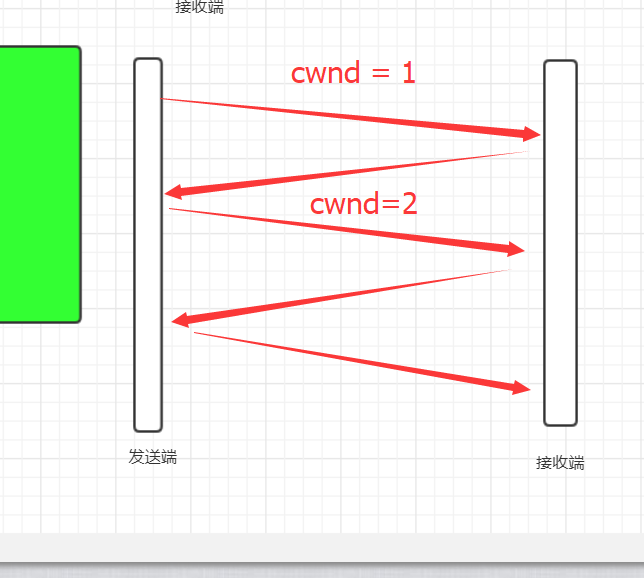


RFC文档里有规定过IP包最小为576个字节。所以说，有很多的设备，MSS=536个字节。

UDP分片，576 – IP头 – udp头 = ？

# 拥塞控制

拥塞窗口：能发送的数据，字节数，cwnd = 1 \* MSS 字节。



慢启动：

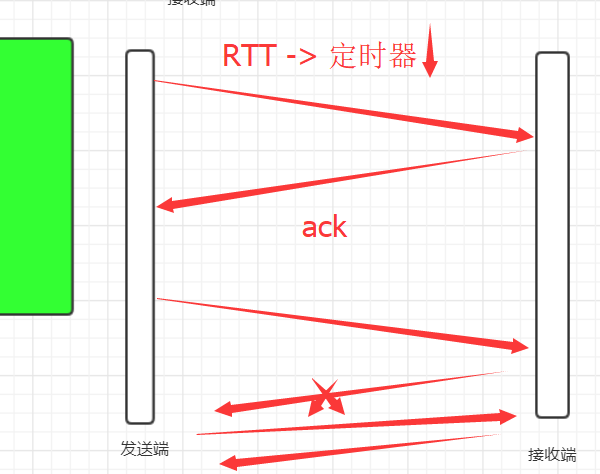
按照一个指数级去增长：1， 2， 4， 8， 16， 32 64 .. 2^n

同样TCP会设置一个阈值，ssthresh = 65535， cwnd >= ssthresh,

会进入拥塞避免：

cwnd = cwnd + 1;

实际上就是把发包的速率降下来了，



拥塞状态：

RENO

1）如果说我重发说丢的包，依然没有收到ACK，或者说收到重发的ACK包超时了，

1. ssthresh = cwnd / 2;
2. cwnd = 1.
3. 进步慢启动

2) 3次收到同样的ACK

1,2,3,4,5,6,7,8,9,10,11,12 🡪 1,2,3,4,5, 7, 8. 10, 12

ack(5)

ack(5)

NEW RENO ack(5)

重传丢失的包，6,7,8,9,10,11,12， --🡪 6, 7, 8, 9

1. ssthresh = cwnd / 2;
2. cwnd = cwnd / 2.
3. 快速恢复的阶段。

CUBIC sack

google 发现或者发明了另外一种拥塞控制算法，BBR。

linux 4.9 版本之后，linux内核采纳了google bbr算法。