# 网络基础之以太网帧,MTU,MSS

<mnstory.net>

## 以太网帧

网络数据传输，底层是物理设备的支持，如路由、交换、网卡、网线，通过网线传送电信号或者光信号，自然可以"连续"地传输，但是，考虑到连续传输，如果中间某个信号被干扰，一个bit不正确会导致整个串都不正确，损失颇大；其次，考虑到线路的多源共享问题，如果一直传送一个源的数据，其他的数据就得不到发送。于是，聪明的设计者将来自一个源的大段数据，拆分成小段，如果某一小段传输不正确，只影响这一小段，而非整体，根据一定的算法交替发送不同来源的数据，可实现线路共享，而分割的这一小段也就是我们说的帧(Frame)，帧也称为协议数据单元(PDU)。

由于帧是逻辑单元，不同的人根据不同需求来指定帧的格式，肯定有所不同，好在目前帧的格式不是很多，具体可参考 <https://wenku.baidu.com/view/25f6f019964bcf84b9d57bb9.html>

简单划分，可以划分为两种：

1. IEEE 802.3 以太网标准，已经过多次更新以便将新技术纳入其中。

2. DIX 以太网标准，现在称其为以太网II(Ethernet II)，是TCP/IP网络中使用的以太网帧格式。

两种标准之间的差异很小，可以用一个格式统一表达，参考<https://en.wikipedia.org/wiki/Ethernet_frame> ：

+-------------------------------+-----------+-------+-----------------+---------------+------------+-----------------------+----------------+----------------------+-----------------+

| | 前序 | 目的Mac | 源MAC | 标签(可选) | 类型或长度 | 数据 | 校验码 | 包间隙 |

+-------------------------------+-----------+-------+-----------------+---------------+------------+-----------------------+----------------+----------------------+-----------------+

| Layer | Preamble | SDF | MAC destination | MAC source | 802.1Q tag |Ethertype (Ethernet II)| Payload | Frame check sequence | Interpacket gap |

| | | | | | (optional) | or length (IEEE 802.3)| | (32?bit CRC) | |

+-------------------------------+-----------+-------+-----------------+---------------+------------+-----------------------+----------------+----------------------+-----------------+

| | 7 octets |1 octet| 6 octets | 6 octets | (4 octets) | 2 octets | 46–1500 octets | 4 octets | 12 octets |

+-------------------------------+-----------+-------+-----------------+---------------+------------+-----------------------+----------------+----------------------+-----------------+

| Layer 2 Ethernet frame | 8 octets | ← 64–1522 octets → | 12 octets |

+-------------------------------+-----------+-------+-----------------+---------------+------------+-----------------------+----------------+----------------------+-----------------+

| Layer 1 Ethernet packet & IPG | ← 72–1530 octets → | 12 octets |

+-------------------------------+-----------+-------+-----------------+---------------+------------+-----------------------+----------------+----------------------+-----------------+

差异表现在：

1. 前序部分，802.3拆分为Preamble(7 octets)和SFD(Start of frame delimiter, 1 octet)；Ethernet II只有Preamble(8 octets)。
2. 对802.3来说，类型或长度部分，表示的是长度，其值小于等于1500；对Ethernet II来说，表示的是类型，其值大于1500，通过值大小即可区分两种帧。

通过上图，我们可以看到，以太网帧主要由三部分组成：

1. 数据Payload部分(46-1500 octets)。

2. 以太网协议本身占用(18-22 octets)，包括：目的Mac(6 octets)、源MAC(6 octets)、标签(可选)(4 octets)、类型或长度(2 octets)、校验码(4 octets)。

3. 以太网协议填充占用(20 octets)，包括：前序(8 octets)、包间隙(12 octets)。

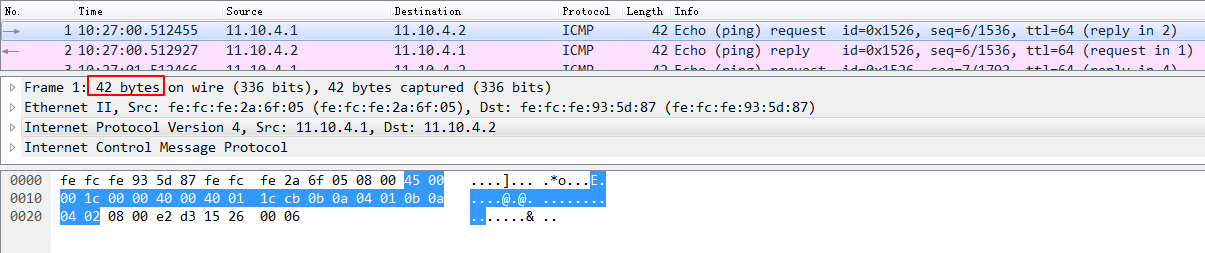
Payload部分，即为每个帧能负载的字节大小。上图表显示Payload为[46, 1500]，但实际测验，低于46 bytes的包也可以传送，不知底层是否做了数据填充：

# ping -s 0 -M do 11.10.4.2

PING 11.10.4.2 (11.10.4.2) 0(28) bytes of data.

8 bytes from 11.10.4.2: icmp\_seq=1 ttl=64

8 bytes from 11.10.4.2: icmp\_seq=2 ttl=64



如上图，整个帧的Payload部分负载只有28 bytes(42-14)，而非46 bytes。

注：我们用ping -s SIZE -M do来发送指定大小的不分片包，理论上，SIZE的最大值为 = 路径最小MTU - IP头部分(20 bytes) - ICMP头部分(8 bytes)。

通常，计算以太网帧的大小，需要算上太网协议本身占用部分，于是，以太网帧的大小范围应该是[64=46+18, 1522=1500+22]，其实最开始EthernetII和802.3规定最大帧为1518，后来为支持vlan，1998年802.3ac规定最大帧为1518+4(vlan)=1522。

先说说，这最小值64 bytes是怎么来的。

以太网通过载波侦听多路访问 (CSMA) 技术控制节点共享访问，CSMA检查介质是否正在传送信号，如果介质上检测到来自另一节点的载波信号，则表示另一设备正在进行传输，这时候会等待，如果没有检查到，会立即传送，但是，也可能检查误差或故障，两设备同时传输，这时候变产生了数据冲突，数据冲突后需要重传这串数据。一条一定长度的线缆，在一串数据发送过程中，如果冲突，需要反馈JAM信号至发送端，这个动作完成前，至少应该这个串的发送过程还没结束，不然到底是哪条数据冲突了都不知道，就没法重传，而个检查反馈的时间，算下来在10Mbps的带宽情况下，刚好需要连续发送64 octets的数据才不至于错过时间。

不过，现在的 LAN 中设备之间几乎所有有线连接都是全双工连接，这类冲突已经不存在。

## MTU

MTU(Maximum Transmission Unit)指的是以太网帧能携带的最大Payload大小，上面已经说过了，是[46, 1500] bytes。

而我们常常设置的，有人叫IP MTU，不过大多数的时候，并没有严格区分。

IP MTU，其确切含义是：一个IP报文不分片能通过某介质(如当前网卡)的最大值。IP报文头部有一个bit(DF, Don't Fragment flag)表示不分片，不分片的报文因为大小限制，可能过不中间的交换设备，这时候会发送ICMP错误回来，设置DF的好处是，路径上的转发设备不再需要分包重组，对连续发送来说，可提高转发效率。

理论上MTU最小值可以是Payload的最小值46 bytes，而你设置IP MTU的时候，并不能设置46 bytes。

先说IPv4，实际上IP MTU最小值(能设置的值)为68 bytes：

# ifconfig eth0 mtu 68

# ifconfig eth0 mtu 67

SIOCSIFMTU: Invalid argument

注：也可以通过 /sys/class/net/eth0/mtu 查看或设置IP MTU。

之所以不能设置为46，是因为IPv4协议有自己的最小值规定，参考 <https://tools.ietf.org/html/rfc791>

Every internet module must be able to forward a datagram of 68 octets without further fragmentation. This is because an internet header may be up to 60 octets, and the minimum fragment is 8 octets.

其实就是，一个IP报文，不管怎么分片，头部不能分片，既然头部的最大值可能为60 bytes，再加上最小分片8 bytes = 68 bytes，所以，IP MTU最小值为68 bytes(最大值是65535 bytes)。

IP MTU还有一个建议值，不一定是最大值，只是从理论、历史、网络环境因素分析，这个值在普遍网络环境下比较适合，比如，IPv4的建议值是1500，前面描述了关于以太网帧结构，应该很清楚了，这1500就是以太网帧的Payload最大值。

网上也有人给出1500的理论分析，因为1542(1522+12+8)这个值，其搬包效率非常高为1500/1542=97.28%，再权衡链路共享，以太网帧校验，延迟，重传长度等因素，就选了这个值，也有一个比较有趣的回答是：

I asked Bob Metcalfe where 1518 bytes comes from. Seriously, I met him at a party many years ago and hit him with this question. His answer, "Hmm, well I really don't know!"

参考 <http://www.mail-archive.com/cisco@groupstudy.com/msg24534.html>

既然特别强调了是IPv4协议，那就还有它的升级版，IPv6。

对IPv6而言，它的最小值是1280 bytes(最大值是65535 bytes，通过jumbogram可以支持4GB大小)，请参考 <https://tools.ietf.org/html/rfc2460>

IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6.

测试一下MTU修改效果：

# ifconfig eth0 mtu 1000

# ifconfig eth0 | grep mtu

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1000

# ping -s 972 -M do 11.10.5.2

PING 11.10.5.2 (11.10.5.2) 972(1000) bytes of data.

980 bytes from 11.10.5.2: icmp\_seq=1 ttl=64 time=0.196 ms

980 bytes from 11.10.5.2: icmp\_seq=2 ttl=64 time=0.717 ms

# ping -s 973 -M do 11.10.5.2

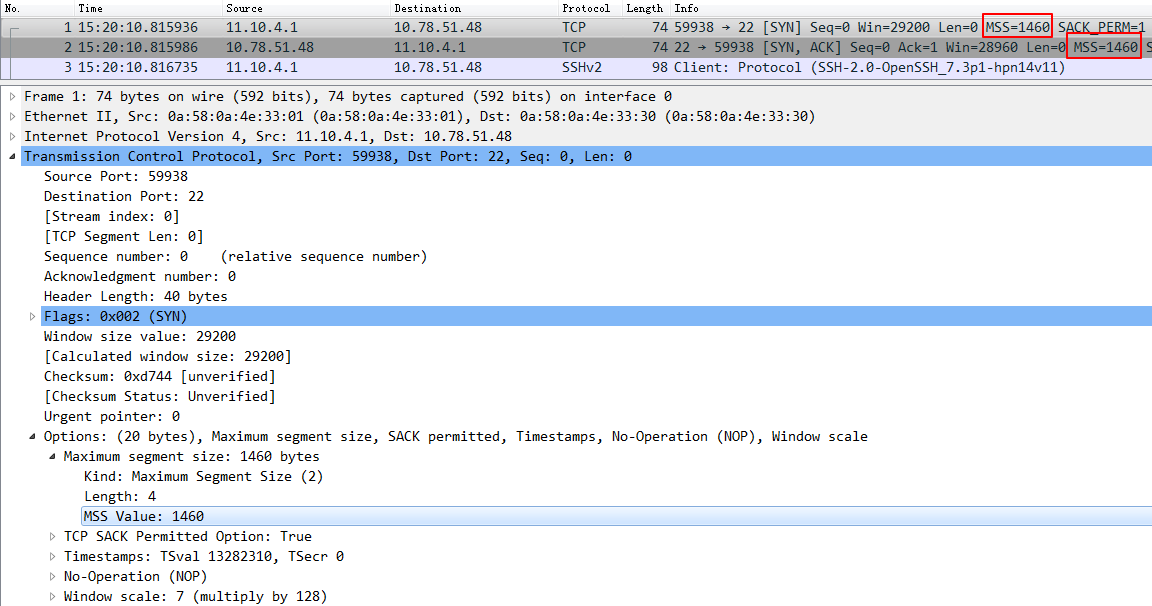
PING 11.10.5.2 (11.10.5.2) 973(1001) bytes of data.

ping: local error: Message too long, mtu=1000

ping: local error: Message too long, mtu=1000

## MSS

先说一下MSS(Maximum Segment Size)最大分段大小，其实是TCP协议层的一个概念，和IP MTU约束IP报文大小类似，MSS约束的是TCP分段大小。在TCP握手的时候可以指定MSS，根据协议可以算出，默认的MSS为：MTU(默认1500 bytes)-IP头(20 bytes)-TCP头(20 bytes) = 1460 bytes。



可以通过iptables设置MSS，例如：

iptables -t mangle -I FORWARD -p tcp --tcp-flags SYN,RST,ACK SYN -j TCPMSS --set-mss 1360 # SYNC 包

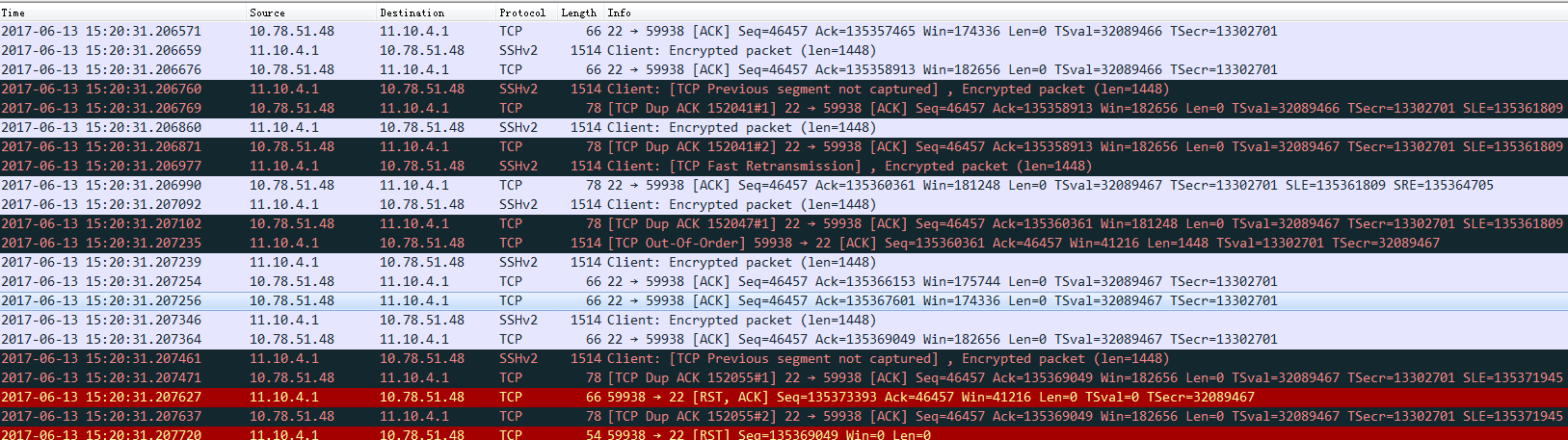
iptables -t mangle -I FORWARD -p tcp --tcp-flags SYN,RST,ACK SYN,ACK -j TCPMSS --set-mss 1360 # SYNC ACK包

如果MSS设置超过MTU，路径上可能返回类似错误：

# tcpdump -i eth0 icmp -nn

10:20:00.986208 IP 11.10.1.2 > 11.10.1.1: ICMP 172.34.0.2 unreachable - need to frag (mtu 1500), length 556

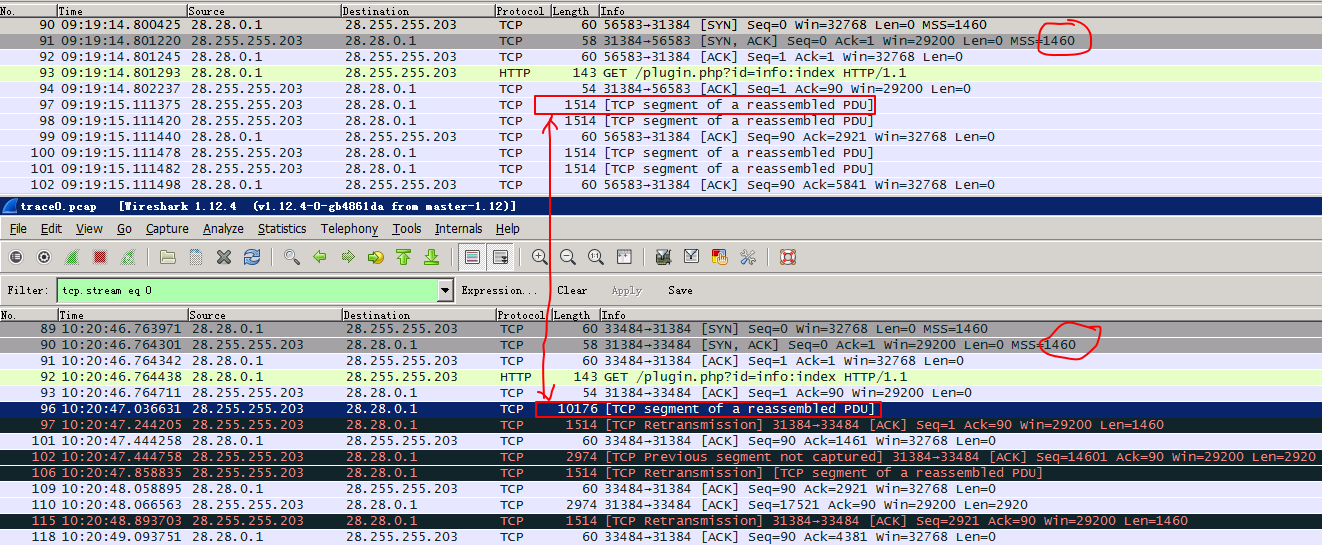
返回类似错误，不一定代表网络不通，小包是没有问题的，UDP一般不受影响，只是TCP表现得性能下降数倍到数十倍，当然，类似现象也有可能是由于分片不当导致，我在QEMU虚拟化环境下(virtio-net)，遇到过此错误(icmp unreachable)，TCP流里面很多重传包，Dup Ack， Out-Of-Order包:



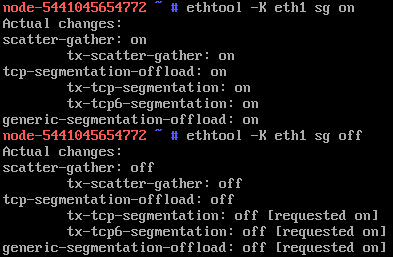
原因并非是MSS和MTU不匹配，而是大包发送端启用了TSO，需要：

ethtool -K eth0 tso off

然后，我又遇到一次在QEMU虚拟化环境下(virtio-net)，这次没有icmp错误返回，但是数据报文正常和异常对比是这样的：



可以看出，异常是从传送了一个大包开始的，虽然指定了MTU，虽然指定了MSS，虽然关闭了tso，然而并没什么卵用，他还是发出了一个大包，最后发现，问题出在回包端的的另外一个参数scatter-gather：



sg参数和tso,gso有关联，关闭的时候，都关闭，开启的时候，都开启。

2017/7/5