## MTU: Maxitum Transmission Unit 最大传输单元 MSS: Maxitum Segment Size 最大分段大小

Ipv4 MTU = MSS - IP包头大小(20)-TCP包头大小(20)

Ipv6 MTU = MSS - IP包头大小(60)-TCP包头大小(20)

## 什么是MTU？

相信大家都见过日常道路上的限高杆，其作用是限制道路上的车辆高度。在计算机网络环境里面，MTU就是链路上的限高杆，与现实世界的道路相似，在网络世界里面不同介质的链路也存在不同的MTU值。



严格来说，虽说单个IP数据包最大长度可达65535字节。而在传输过程中，传输链路规定了单个数据包传输长度。当IP数据包长度超过此长度后，数据包将被切割为小于或等于链路规定长度的小IP包。

此规定长度就是MTU值，全名为：Maximum Transmission Unit 最大传输单元。

**（1）数据报大小**IPv4的数据报最大大小是65535字节，包括IPv4首部。因为首部中说明大小的字段为16位。  
IPv6的数据报最大大小是65575字节，包括40字节的IPv6首部。同样是展16位，但是IPv6首部大小不算在里面，所以总大小比IPv4大一个首部（40字节）。

**（2）MTU**许多网络有一个可由硬件规定的MTU。以太网的MTU为1500字节。有一些链路的MTU的MTU可以由认为配置。IPv4要求的最小 链路MTU为68字节。这允许最大的IPv4首部（包括20字节的固定长度部分和最多40字节的选项部分）拼接最小的片段（IPv4首部中片段偏移字段以 8个字节为单位）IPv6要求的最小链路MTU为1280字节。

**（3）分片（fragmentation）**当一个IP数据报从某个接口送出时，如果它的大小超过相应链路的MTU，IPv4和IPv6都将执行分片。这些片段在到达终点之前通常 不会被重组（reassembling）。IPv4主机对其产生的数据报执行分片，IPv4路由器则对其转发的数据报进行分片。然后IPv6只有主机对其 产生的数据报执行分片，IPv6路由器不对其转发的数据报执行分片。

IPv4首部的“不分片”（do not fragment）位（即DF位）若被设置，那么不管是发送这些数据报的主机还是转发他们的路由器，都不允许对它们分片。当路由器接收到一个超过其外出链 路MTU大小且设置了DF位的IPv4数据报时，它将产生一个ICMPv4“destination unreachable,fragmentation needed but DF bit set”（目的不可到达，需分片但DF位已设置）的出错消息。

既然IPv6路由器不执行分片，每个IPv6数据报于是隐含一个DF位。当IPv6路由器接收到一个超过其外出链路MTU大小的IPv6数据报时， 它将产生一个ICMPv6 “packet too big”的出错消息。IPv4的DF位和隐含DF位可用于路径MTU发现。

**（4）最小重组缓冲区大小（minimum reassembly buffer size）**IPv4和IPv6都定义了最小缓冲区大小，它是IPv4或IPv6任何实现都必须保重支持的最小数据报大小。其值对IPv4为576 字节，对于IPv6为1500字节。例如，对于IPv4而言，我们不能判定某个给定的目的能否接受577字节的数据报，为此很多应用避免产生大于这个大小 的数据报。

**（5）MSS（maximun segment size）**TCP有一个最大分段大小，用于对端TCP通告对端每个分段中能发送的最大TCP数据量。MSS的目的是告诉对端其重组缓冲区大小的实 际值，从而避免分片。MSS经常设计成MTU减去IP和TCP首部的固定长度。以太网中使用IPv4MSS值为1460，使用IPv6的MSS值为 1440（两者TCP首部都是20字节，但是IPv6首部是40字节，IPv4首部是20字节）。

**（6）TCP发送缓冲区**每个TCP套接字有一个发送缓冲区，我们可以用SO\_SNDBUF套接字选项来更改该缓冲区的大小。当某个应用进程调用write时， 内核从该应用进程的缓冲区复制所有数据到缩写套接

字的发送缓冲区。如果该套接字的发送缓冲区容不下该应用进程的所有数据（或是应用进程的缓冲区大于套接字 的发送缓冲区，或是套接字的发送缓冲区中已有其他数据），该应用进程将被投入睡眠。这里假设该套接字是阻塞的，它通常是默认设置。内核将不从write系 统调用返回，直到应用进程缓冲区中的所有数据都复制到套接字发送缓冲区。因此，从写一个TCP套接字的write调用成功返回仅仅表示我们可以重新使用原 来的应用进程缓冲区，并不表明对端的TCP或应用进程已接受到数据。

这一端的TCP提取套接字发送缓冲区中的数据并把它发送给对端的TCP，其过程基于TCP数据传送的所有规则。对端TCP必须确认收到的数据，伴随 来自对端的ACK的不断到达，本段TCP至此才能从套接字发送缓冲区中丢弃已确认的数据。TCP必须为已发送的数据保留一个副本，直到它被对端确认为止。 本端TCP以MSS大小或是更小的块把数据传递给IP，同时给每个数据块安上一个TCP首部以构成TCP分节，其中MSS或是由对端告知的值，或是 536（若未发送一个MSS选项为576-TCP首部-IP首部）。IP给每个TCP分节安上一个IP首部以构成IP数据报，并按照其目的的IP地址查找 路由表项以确定外出接口，然后把数据报传递给相应的数据链路。每个数据链路都有一个数据队列，如果该队列已满，那么新到的分组将被丢弃，并沿协议栈向上返 回一个错误：从数据链路到IP，在从IP到TCP。TCP将注意到这个错误，并在以后某个时候重传相应的分节。应用程序不知道这种暂时的情况。

**（7）UDP发送缓冲区**任何UDP套接字都有发送缓冲区大小（我们可以用SO\_SNDBUF套接字选项更改它），不过它仅仅是可写道套接字UDP数据报大小上 限。如果一个应用进程写一个大于套接字发送缓冲区大小的数据报，内核将返回该进程一个EMSGSIZE错误。既然UDP是不可靠的，它不必保存应用进程数 据的一个副本，因此无需一个真正的发送缓冲区。（应用进程的数据在沿协议栈向下传递时，通常被复制到某种格式的一个内核缓冲区中，然而当该数据被发送之 后，这个副本被数据链路层丢弃了。）

UDP简单地给来自用户的数据报安上8字节首部以构成UDP数据报，然后传递给IP。IPv4或IPv6给UDP数据报安上相应的IP首部以构成 IP数据报，执行路由操作确定外出接口，然后或者直接把数据报加入数据链路层输出队列（如果适合于MTU），或者分片后在把每个片段加入数据集链路层的输 出队列。如果某个UDP进程发送大数据报，那么它们相比TCP应用数据更有可能被分片，因为TCP会把应用数据划分成MSS大小的块，而UDP却没有对等 的手段。

从写一个UDP套接字的write调用成功返回表示所写的数据报或其所有片段已被加入数据链路层的输出队列。如果该队列没有足够的空间存放该数据报 或它的某个片段，内核通常会返回一个ENOBUFS错误给它的应用进程。有些UDP实现不返回这种错误，这样甚至数据报未经发送就被丢弃的情况进程也不知 道