# 网络基础之TSO,UFO,GSO,LRO,GRO

<mnstory.net>

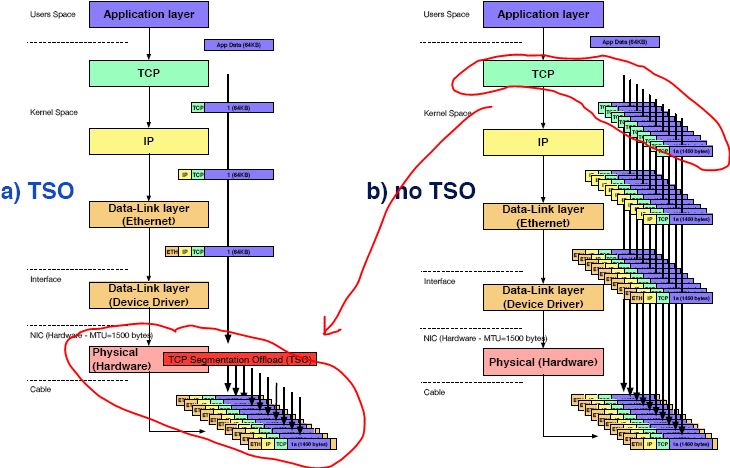
本文原创部分占比不多，图片多是盗用，不知道最开始是谁画的，我是从<http://geek.csdn.net/news/detail/67260> 取的。此文主要是做一个自己的梳理。

## TSO/UFO

TCP的每个数据包大小不能超过MSS值，在发送端超过的，需按照MSS进行分段(segment)。

分段这事情，老是让CPU来做，那是很耗精力的事，做网卡的乐意分享此任务，将其offload到网卡，当网卡支持TCP分段这功能的时候，我们就说它支持TSO(TCP segmentation offload)。

如果网卡不支持TSO功能，TCP数据会在内核协议栈的TCP层按照MSS分成若干段(segmentation)，然后经IP层，经链路层，经设备驱动层，再达网卡。有了TSO功能，在TCP大数据块不超过64K的情况下，可以一个大包走到网卡。对比一下，两者的差异：



图：有tso vs 没有tso

有TSO，内核协议栈上少了很多数据包的划分、小包传输、处理、checksum、上下文切换等，从而减轻了CPU的任务。

IP分片后，只有第一个分片包带有上层协议的头部(例如，UDP头部、ICMP头部)其余分片只有IP头。当IP分片到达最终目的地后，根据IP头部中的信息，在网络层进行重组，而不会在路径上重组。

TCP分段，会在每个分段上都附上TCP的协议头，到达端点后，根据TCP头部信息在传输层进行重组。如果一个TCP分段需要多个IP分片来承接，那么当其中一个IP分片丢失，这个TCP分段将重传(意味着多个IP分片重传)，IP层本身是不可靠的，不具备重传机制，需要TCP层来进行，所以，应该尽量避免一个TCP分段需要多个IP分片来承接，默认MSS根据MTU来计算的，也正是这个原因。

UDP协议本身不具备类似TCP的分段功能，所以，大的UDP数据报文，需要由IP层来进行分片传送，和TSO类似，网卡将UDP报文的IP分片功能offload到网卡来做，那就叫**UFO**(udp-fragmentation-offload)技术。

查看一下，网卡是否支持TSO/UFO：

# ethtool -k eth0

tcp-segmentation-offload: on

tx-tcp-segmentation: on

tx-tcp-ecn-segmentation: off [fixed]

tx-tcp6-segmentation: on

udp-fragmentation-offload: off [fixed]

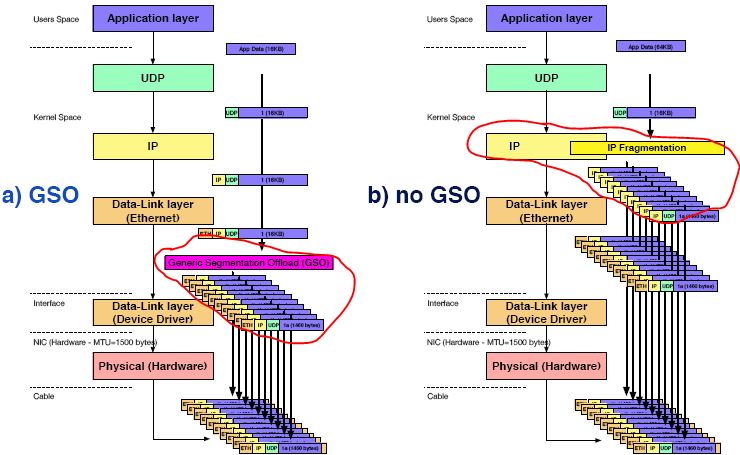
## GSO

就TSO功能而言，它的主要优点有二：

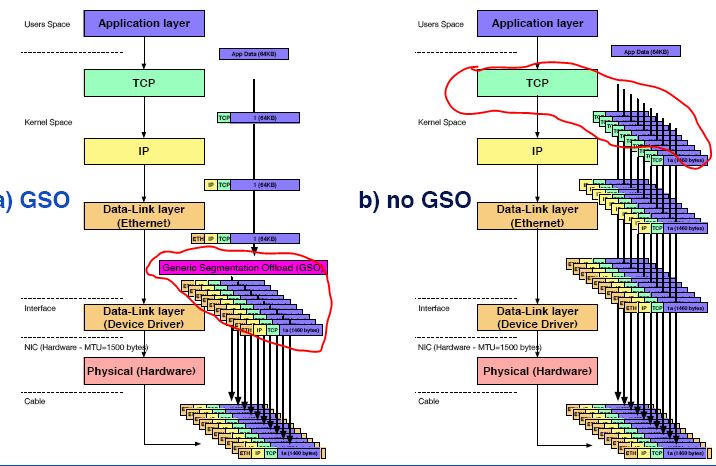
1. TCP/IP协议栈上传送一次大块数据，而非多次小块数据，节约CPU。

2. 让网卡来分段功能，节约CPU，这点需要硬件支持。

如果硬件不支持，我们就做不了TSO，但是，我们可以做GSO(Generic Segmentation Offload)，GSO将TSO功能泛化，让数据跨过IP层，链路层，在数据离开协议栈，进入网卡驱动前进行分段，不论是TCP还是UDP，都是分段(每个包都附加TCP/UDP头部)，这样，当一个段丢失，不需要发送整个TCP/UDP报文。其次，路径上的CPU消耗也会减少，所以可以说GSO是对UFO的一种改良(UFO是IP分片，只有头包带有UDP头，丢掉一片要整段重传)



图：对于 UDP，在物理网卡不支持 UFO 时，使用和不使用 GSO 的情形



图：对 TCP，在网卡不支持 TSO 时，使用和不使用 GSO 的情形

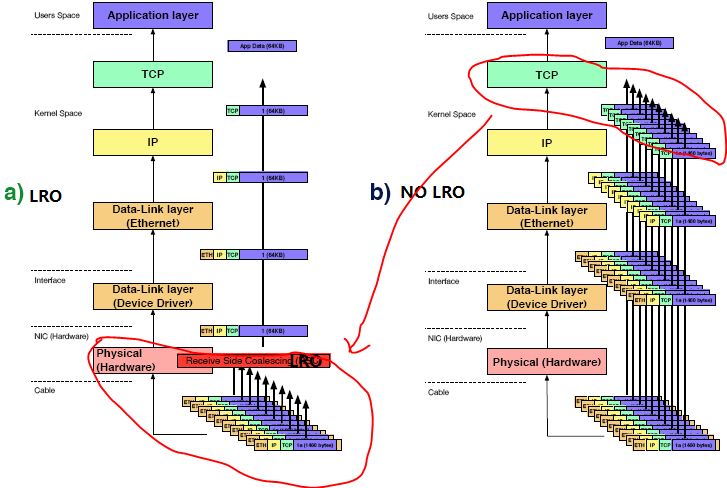
查看一下，GSO是否开启：

# ethtool -k eth0

generic-segmentation-offload: on

## LRO

发包的过程说完，接收TCP包的时候，如果网卡能将多个TCP分段合并成一个超级SKB，然后递交到上层网络，以减少CPU消耗，这个就叫网卡支持LRO(large receive offload)。



图：物理网卡支持 LRO vs不支持LRO

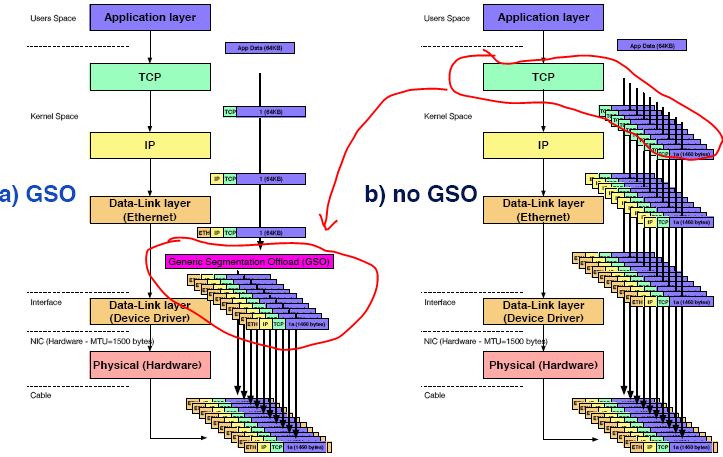
查看一下，网卡是否支持LRO：

# ethtool -k eth0

large-receive-offload: off [fixed]

## GRO

LRO本身实现上有一些问题，比如多个包的状态信息不一致，合并后会导致状态被破坏，并且也依赖于网卡。于是有了更通用的版本GRO(generic-receive-offload)，他将合并操作放到了设备启动层，保留了每个数据包的一些熵信息(五元组)供后面解析使用。



图：在不支持 LRO 的情况下，对 TCP 使用和不使用 GRO 的情形

**Offload 传输段还是接收端 针对的协议 Offloading 的位置 ethtool 命令输出中的项目**

TSO 传输段 TCP NIC tcp-segmentation-offload 需要网卡支持，多数都支持

UFO 传输段 UDP NIC udp-fragmentation-offload 需要网卡支持，多数不支持

GSO 传输段 TCP/UDP NIC或者离开IP协议栈进入网卡驱动前 generic-segmentation-offload 内核特征，主要是延迟分段

LRO 接收段 TCP NIC large-receive-offload 需要硬件支持，多数都支持

GRO 接收段 TCP/UDP NIC或者离开网卡驱动进入IP协议栈前 generic-receive-offload 内核特征，主要是提前合并

查看一下，GRO是否开启：

# ethtool -k eth0

generic-receive-offload: on

2017/7/10