Checksum Offloads

Introduction

This document describes a set of techniques in the Linux networking stack to take advantage of checksum offload capabilities of various NICs.

The following technologies are described:

- TX Checksum Offload
- LCO: Local Checksum Offload
- RCO: Remote Checksum Offload

Things that should be documented here but aren't yet:

- RX Checksum Offload
- CHECKSUM_UNNECESSARY conversion

TX Checksum Offload

The interface for offloading a transmit checksum to a device is explained in detail in comments near the top of include/linux/skbuff.h.

In brief, it allows to request the device fill in a single ones-complement checksum defined by the sk_buff fields skb->csum_start and skb->csum_offset. The device should compute the 16-bit ones-complement checksum (i.e. the 'IP-style' checksum) from csum_start to the end of the packet, and fill in the result at (csum_start + csum_offset).

Because csum_offset cannot be negative, this ensures that the previous value of the checksum field is included in the checksum computation, thus it can be used to supply any needed corrections to the checksum (such as the sum of the pseudo-header for UDP or TCP).

This interface only allows a single checksum to be offloaded. Where encapsulation is used, the packet may have multiple checksum fields in different header layers, and the rest will have to be handled by another mechanism such as LCO or RCO.

CRC32c can also be offloaded using this interface, by means of filling skb->csum_start and skb->csum_offset as described above, and setting skb->csum_not_inet: see skbuff.h comment (section 'D') for more details.

No offloading of the IP header checksum is performed; it is always done in software. This is OK because when we build the IP header, we obviously have it in cache, so summing it isn't expensive. It's also rather short.

The requirements for GSO are more complicated, because when segmenting an encapsulated packet both the inner and outer checksums may need to be edited or recomputed for each resulting segment. See the skbuff.h comment (section 'E') for more details.

A driver declares its offload capabilities in netdev->hw_features; see Documentation/networking/netdev-features.rst for more. Note that a device which only advertises NETIF_F_IP[V6]_CSUM must still obey the csum_start and csum_offset given in the SKB; if it tries to deduce these itself in hardware (as some NICs do) the driver should check that the values in the SKB match those which the hardware will deduce, and if not, fall back to checksumming in software instead (with skb_csum_hwoffload_help() or one of the skb_checksum_help() / skb_crc32c_csum_help functions, as mentioned in include/linux/skbuff.h).

The stack should, for the most part, assume that checksum offload is supported by the underlying device. The only place that should check is validate_xmit_skb(), and the functions it calls directly or indirectly. That function compares the offload features requested by the SKB (which may include other offloads besides TX Checksum Offload) and, if they are not supported or enabled on the device (determined by netdev->features), performs the corresponding offload in software. In the case of TX Checksum Offload, that means calling skb_csum_hwoffload_help(skb, features).

LCO: Local Checksum Offload

LCO is a technique for efficiently computing the outer checksum of an encapsulated datagram when the inner checksum is due to be offloaded.

The ones-complement sum of a correctly checksummed TCP or UDP packet is equal to the complement of the sum of the pseudo header, because everything else gets 'cancelled out' by the checksum field. This is because the sum was complemented before being written to the checksum field.

More generally, this holds in any case where the 'IP-style' ones complement checksum is used, and thus any checksum that TX Checksum Offload supports.

That is, if we have set up TX Checksum Offload with a start/offset pair, we know that after the device has filled in that checksum, the ones complement sum from csum_start to the end of the packet will be equal to the complement of whatever value we put in the checksum field beforehand. This allows us to compute the outer checksum without looking at the payload: we simply stop summing when we get to csum_start, then add the complement of the 16-bit word at (csum_start + csum_offset).

Then, when the true inner checksum is filled in (either by hardware or by skb_checksum_help()), the outer checksum will become

correct by virtue of the arithmetic.

LCO is performed by the stack when constructing an outer UDP header for an encapsulation such as VXLAN or GENEVE, in udp_set_csum(). Similarly for the IPv6 equivalents, in udp6_set_csum().

It is also performed when constructing an IPv4 GRE header, in net/ipv4/ip_gre.c:build_header(). It is *not* currently performed when constructing an IPv6 GRE header; the GRE checksum is computed over the whole packet in net/ipv6/ip6_gre.c:ip6gre_xmit2(), but it should be possible to use LCO here as IPv6 GRE still uses an IP-style checksum.

All of the LCO implementations use a helper function lco_csum(), in include/linux/skbuff.h.

LCO can safely be used for nested encapsulations; in this case, the outer encapsulation layer will sum over both its own header and the 'middle' header. This does mean that the 'middle' header will get summed multiple times, but there doesn't seem to be a way to avoid that without incurring bigger costs (e.g. in SKB bloat).

RCO: Remote Checksum Offload

RCO is a technique for eliding the inner checksum of an encapsulated datagram, allowing the outer checksum to be offloaded. It does, however, involve a change to the encapsulation protocols, which the receiver must also support. For this reason, it is disabled by default.

RCO is detailed in the following Internet-Drafts:

- https://tools.ietf.org/html/draft-herbert-remotecsumoffload-00
- https://tools.ietf.org/html/draft-herbert-vxlan-rco-00

In Linux, RCO is implemented individually in each encapsulation protocol, and most tunnel types have flags controlling its use. For instance, VXLAN has the flag VXLAN_F_REMCSUM_TX (per struct vxlan_rdst) to indicate that RCO should be used when transmitting to a given remote destination.