INTERNET-DRAFT
Intended Status: Informational

T. Herbert
Facebook

February 29, 2016

Remote checksum offload for VXLAN
draft-herbert-vxlan-rco-01

Abstract

This specification describes remote checksum offload for VXLAN. Remote checksum offload is a mechanism that provides checksum offload of transport checksums in encapsulated packets using rudimentary offload capabilities found in most Network Interface Card (NIC) devices. The outer UDP checksum is enabled on transmit and, with some additional meta data, a receiver is able to deduce the checksum to be set in an encapsulated packet. Effectively this offloads the computation of the inner checksum which can be a significant performance optimization. Enabling the UDP checksum has the additional advantage that it covers more of the packet including the IP pseudo header and virtual network identifier.

Table of Contents

1 Introduction

Remote checksum offload is a mechanism that uses rudimentary NIC offload features to support offloading checksum calculation of encapsulated packets. The background and motivation for remote checksum offload is presented in [RCO].

In this specification we describe remote checksum offload for VXLAN [RFC7348]. In this design the UDP [RFC0768] checksum is enabled on transmit, and optional data conveyed in the VXLAN header specifies the location of the checksum field being offloaded and its starting point for computation. Upon receipt, after the UDP checksum is verified, the receiver sets the offloaded checksum field per the computed packet checksum and the data in the header.

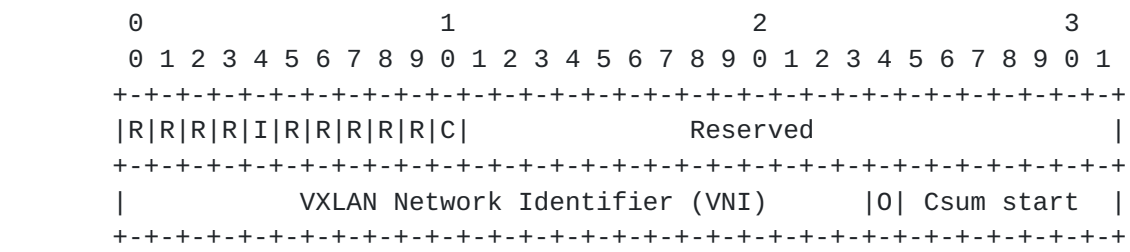
This design should also be compatible with VXLAN-GPE [VXLANGPE].

2 Remote checksum offload for VXLAN

This section describes remote checksum offload for VXLAN.

2.1 Header format

VXLAN header with remote checksum data:



- C bit: Remote checksum offload bit. When set indicates that the remote checksum offload data is present.

- O bit: Offset bit. Indicates the checksum offset relative to checksum start. Two offsets are supported corresponding to TCP [RFC0793] and UDP [RFC0768].

  O = 1 indicates checksum offset is checksum start + 6 (UDP)

  O = 0 indicates checksum offset is checksum start + 16 (TCP)

- Csum start: Checksum start divided by two. Checksum start is relative to the the first byte of the encapsulated packet. Note that only even offsets are supported and that the maximum value is 254. This typically refers to the offset of a transport

header.

The remote checksum data is encoded within the eight reserved bits of the VXLAN header that follow the VNI. A flag bit is allocated to indicate the presence of the remote checksum data.

2.2 Transmitter operation

The typical actions to set remote checksum offload on transmit are:

1) Transport layer creates a packet and indicates in internal packet meta data that checksum is to be offloaded to the NIC (normal transport layer processing for checksum offload). The checksum field is populated with the bitwise "not" of the checksum of the pseudo header.

2) VXLAN header is added to the packet to do encapsulation. If the transport checksum is for UDP or TCP, checksum start is even, and checksum start relative to start of the payload is <= 254, then remote checksum offload may be used. To set remote checksum offload the C bit is set, the O bit is set for a UDP offset or cleared for a TCP offset, and checksum start value divided by two is set in the csum start field.

3) Encapsulation layer arranges for NIC checksum offload of the outer UDP header checksum. This supersedes the settings to offload the inner packet's transport checksum.

4) Packet is sent to the NIC. The NIC will perform transmit checksum offload and set the checksum field in the outer UDP header. The inner header and rest of the packet are transmitted without modification.

2.3 Receiver operation

The typical actions a VXLAN receiver does to support remote checksum offload are:

1) Receive packet and validate outer checksum following normal processing (ie. validate non-zero UDP checksum).

2) Deduce full checksum for the IP packet. This is directly provided if a device returns the packet checksum in checksum-complete or checksum-unnecessary conversion can be done.

3) If the C bit is set, remote checksum offload is enabled. Checksum start is csum start value times two. If O bit is set then checksum offset is checksum start + 6, else it is checksum

start + 16.

4) From the packet checksum, subtract the checksum computed from the start of the packet (outer IP header) to the offset in the packet indicted by checksum start. The result is the deduced checksum to set in the checksum field of the encapsulated transport packet.

5) Write the resultant checksum value into the packet at the offset provided by checksum offset.

6) Adjust the saved packet checksum to account for changing the checksum field within the packet.

7) Checksum is verified at the transport layer using normal processing. This should not require any checksum computation over the packet since the complete checksum has already been provided.

Steps 3,4,5, and 6 in pseudo code:

```
packet_csum: checksum computed by receiver covering the start
  of the packet (outer IP header) to the end of the packet

start_of_packet: memory address of start of packet

offset_encap_payload: offset of encapsulation payload relative
  to start_of_packet

csum_start: value of csum start field

o_bit: value of the O bit

checksum(start, len): function to compute checksum from start
  address for len bytes

// Derive the start and offset values
start = csum_start * 2
if (o_bit)
    offset = start + 6
else
    offset = start + 16

// Compute packet checksum starting from checksum start value
// (1's complement arithmetic)
csum = packet_csum - checksum(start_of_packet,
                              offset_encap_payload + start)
```

```
// Set derived checksum in the checksum field
old = *(start_of_packet + offset_encap_payload + offset)
*(start_of_packet + offset_encap_payload + offset) = csum

// Adjust packet checksum (1's complement arithmetic)
packet_csum += (csum - old)
```

3 Security Considerations

Remote checksum offload should not impact protocol security.

4 IANA Considerations

There are no IANA considerations in this specification. Remote checksum offload requires a one VXLAN reserved bit and use of the eight reserved bits after the VNI.

5 References

5.1 Normative References

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

5.2 Informative References

[RCO] Herbert T., "Remote checksum offload", draft-herbert-remotecsumoffload-02.

[VXLANGPE] Quinn P. and et al., "Generic Protocol Extension for VXLAN", draft-quinn-vxlan-gpe-04.txt

Authors' Addresses

Tom Herbert
Facebook
1 Hacker Way
Menlo Park, CA

US

EMail: tom@herbertland.com