Tunneling (&) Offloads

David S. Miller
Red Hat Inc.
Netfilter Workshop 2016, Amsterdam

What Is Tunneling

Basically, any form of encapsulation.

Use cases:

- Routing private addresses over a public network
- Routing protocol X over protocol Y
- Layering a L2 fabric over L3 infrastructure (virtualization, clouds)
- Encryption and authentication (VPNs, IPSEC, etc.)

Examples: GRE, L2TP, VXLAN, MACVLAN

Tunneling is quite pervasive today.

UDP Tunnels

Extremely “friendly” manner to encapsulate

Can leverage existing hardware checksum offloads

Allows Receive Side Scaling to still occur (UDP source port is set to a hash representing the inner packet’s flow)

UDP ports provide a built-in demuxing mechanism

Generic UDP Encapsulation (GUE)

GUE header sits between UDP header and encapsulated packet

The Problem With UDP Ports

Existing VXLAN offloading uses simplistic keys.

It only matches on the UDP destination port.

Even a full lookup key using src and dst addresses is not enough.

Consider PFs and VFs split across multiple VMs.

Traffic on host A for port X might be VXLAN…

… but traffic on host B for port X might not. Hardware can’t tell.

Transports Over UDP

Goals are laudable:

- Protocol advancements propagate more quickly
- Middleware boxes are taken out of the equation

Implementation is somewhat scary, protocols implemented in userspace and are encapsulated via GUE, with optional encryption of payload

These things exist already with different names: QUIC

How backlogged are protocol features? Android doesn’t support TFO yet.

UDP Tunnels And Checksumming

On the surface, it looks like we potentially have 2 checksums

One for inner TCP packet, and one for encapsulating UDP frame

Problematic because HW typically supports one checksum offload

Checksum of outer UDP packet is actually constant

We can thus now do “local checksum offload”, from Edward Cree

Why? Once checksum field is filled in, inner TCP packet’s checksum is “zero”

Receive Checksum Offload

Mostly obviated by local checksum offload, but might still be useful

Hardware only offloads checksum of encapsulating protocol

Encapsulated frame is left not checksummed

Metadata added to GUE header helps receiver deduce inner checksum

It’s like local checksum offload, but in the opposite direction

UDP Tunnels and GSO

Another hardware offload potentially lost with tunnels

Segmentation offload engines want two things:

- Pointer to IP header and it’s size
- Pointer to inner TCP frame

Trick is to give the HW an IP header length which includes the tunnel headers

Called “Partial Segmentation Offload”, from Alexander Duyck

GRO/LRO for Tunnels, Briefly...

This is a non-trivial area.

Hardware aggregation has several problems:

- The packet is mangled (cannot reconstitute original packet stream)
- No control over aggregation heuristics

Possible future solutions: BPF aggregation engines