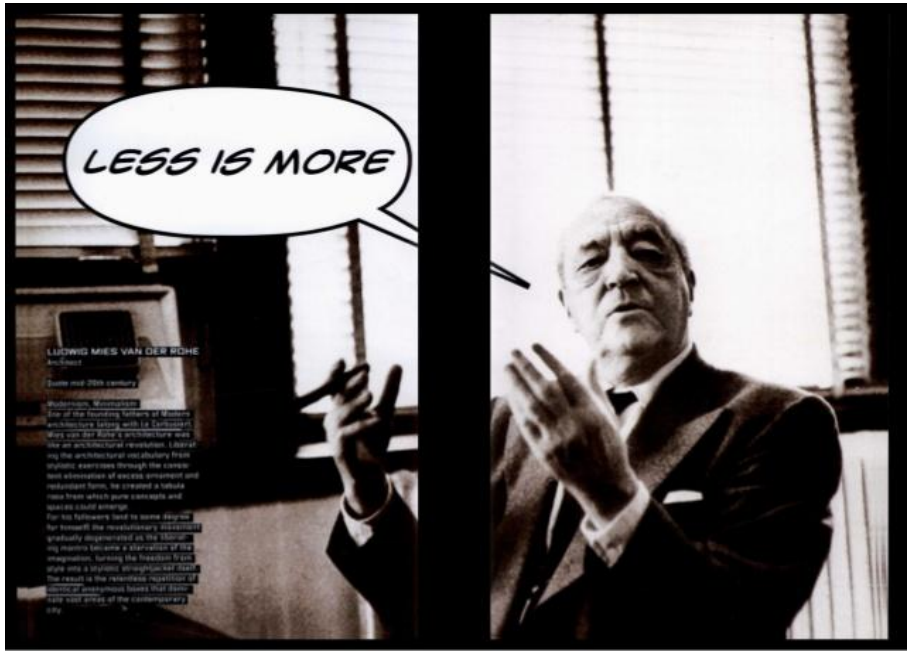
Hardware Checksumming



David S. Miller
Red Hat Inc.


If I could do less, I would do that more often.

Overview

- Internet checksumming basics
- Hardware History
- Tunneling and Encapsulation
- Arguments for a ubiquitous 1’s complement checksum

Internet Checksumming Basics

1. 16-bit 1's complement sum
2. Computed with checksum field of header set to zero
3. For protocols (UDP, TCP, etc.) 'pseudo header' is included
4. Many arithmetic tricks exist to speed up calculation
   a. Add with carry instructions
   b. Folding

Hardware History: Part 1

Dumb stateless devices, no hardware checksumming offload.

Ring buffers, if you were lucky.

Checksumming *“for free”* with checksum+copy operations.

Trivial for send, less trivial but doable for receive (run the TCP stack in recvmsg() control flow).

Hardware History: Part 2

Scatter-gather and checksumming offloads, but *“stateful”*.

On receive it only gives a boolean state, saying whether the checksum is good or bad.

Limited to protocol headers the parser engine understands. New protocol or tunnel encapsulation? New hardware.

Not extensible. (... time in place solution …)

Hardware History: Part 2.1

Chip provides 1’s complement sum computed over entire RX packet contents.

Stack can cheaply adjust this sum as it pulls headers.

Works for arbitrary tunneling technologies and layering.

On TX, chip is told the point at which to start computing the checksum, and where to place the 16-bit result. Checksum field is pre-seeded with pseudo header checksum.

Tunneling and Encapsulation

1. Multiple checksums
2. Which one, if any, to offload?
3. In the presence of multiple checksums, what does skb->csum_start and skb->csum_offset get set to?
4. Ironically, we don’t actually need support for offloading multiple checksums.

Tunneling Part 2

Edward Cree's observation *"local checksum offload"*:

Outer checksum does not depend upon full payload.

Why?

1's complement sum is weak, don't make it weaker.

On fwd encapsulation we checksum full outer packet.

Tunneling Part 3: Remote Checksum Offload

From Tom Herbert, must be supported by both ends

Metadata describes how to deduce the inner checksum

Only the outer checksum is performed on transmit

Receiver validates outer checksum, decapsulates, then uses the metadata to deal with inner checksum.

Mostly obviated by local checksum offload, but not completely.

Protocol Ossification

Too finely focused hardware offloads have a terrible side-effect

Several aspects of the internet become set in stone

If it's not UDP or TCP, the facility is unusable

Hence proliferation of *'foo-over-UDP'* tunneling technologies and ideas which will destroy the internet, such as those involving using TCP for tunnel encapsulation

1’s Complement CSUM Everywhere

1. Less complex logic in chipsets.
2. More consistent feature sets and performance
3. The kernel becomes simpler
4. Supporting new *“stuff”* is easier, less ossification
5. In large to huge data centers, the lowest common denominator is what matters.

Message to hardware designers: Raw 1’s complement offloads are what we want

Checksum Conclusions

The kernel’s needs are simple and minimal.

Sophisticated hardware checksum facilities have little to no value at all

In fact, they get in the way

Raw 1’s complement facilities allow us to support anything


What part of “less is more” didn’t you understand?

The State of Switchdev

“And in advance, I’d like to thank the first hardware vendor to merge hw switching driver upstream. You will be a true trail-blazer.” -Ottawa 2015

Thanks to Mellanox, someone answered the call.

Tell your hardware vendor that SWITCHDEV based Linux solutions are what you want from them!

Thanks

Pablo Neira Ayuso and the whole netdevconf team

Tom Herbert for beating the checksumming drum for the past few years

Hardware driver maintainers for

listening and giving feedback