Encapsulation and networking offload support

Networking workshop @LKS
August 19, 2014

Tom Herbert <therbert@google.com>

Topics

- Prospects/requirements
- Essential offloads
  - Packet steering
  - RX/TX checksum offload
  - TSO/GSO
  - LRO/GRO
- Other offloads

Encapsulation cases

- Non-virtualization
  - GRE, MPLS, IPIP, SIT
  - Source routing
  - IPv6 encap over IPv4 infrastructure
  - L2,L3 over L3
  - L4/L3 (e.g. QUIC, ESP/UDP)
  - DC encryption on the horizon
- Virtualization
  - VXLAN, nvgre (GUE, geneve, LISP, L2TP)
  - Overlay networks, isolation for third party guests
  - “internal” customers as tenants
  - L2,L3 over L3
  - Virtual network ID is critical value

Requirements of networking stack

- Efficient data path that provides security, performance, and control
- Good solution should not *require* any special, encapsulation protocol-aware HW
  - Advanced HW may augment stack functions
- Core kernel paths should be encap-protocol agnostic also
  - Encap might not even be kernel based

UDP based encapsulation

- UDP works with existing HW infrastructure
  - RSS in NICs, ECMP in switches
  - Checksum offload
- Used in nearly all encap, NV data protocols
  - VXLAN, LISP, MPLS, GUE, Geneve, NSH, L2TP
- Likelihood UDP based encapsulation (aka foo/UDP) becomes ubiquitous
  - In time most packets in DC could be UDP!

Packet steering

- Want flow hash corresponding to *inner packet*
  - Host, NIC, switches are interested in this
  - RSS/ECMP/RPS/RFS/aRFS
- Q: How to avoid HW needing to do deep parsing into encapsulation?

Steering for UDP encapsulation

- Leverage existing UDP flow hash
- Sender: set UDP to source port to hash of inner packet (**udp_flow_src_port**)
- Receiver: 5-tuple hash over outer headers provides a flow hash of inner packet
- Need to start enabling UDP RSS on devices for this to get most benefit!

Steering for non-UDP

- Encapsulation over IPv6
  - Flow label set to hash (auto_flowlabels, sockopt)
  - Receiver uses flow label as “ports” in 5-tuple hash calculation
- GRE
  - Need to perform deep parsing
  - keyID might be sufficient
- Other protocols
  - Deep parsing to find inner L4 (doesn’t always work)
  - Use something in encap header resembling GRE keyID or flow label

Checksum offload

- Possibility of two or even three checksums in a single packet
  - e.g. IP->**UDP->GRE->**IP->**TCP**
- Note that outer checksums protect more and potentially cover inner checksums
- But, switch vendors are pushing to avoid using checksums in UDP encapsulation

Receive checksum overhaul

- Prerequisite to checksum/encapsulation
- Goals
  - Define what CHECKSUM_UNNECESSARY means
  - Preserve CHECKSUM_COMPLETE through encapsulation layers
  - Don’t do skb_checksum more than once per packet
  - Make GRO/non-GRO csum processing consistent
  - Unified set of stanard checksum functions
- Almost done! :-)

RX checksum offload

- CHECKSUM_COMPLETE
  - Always works, any number of checksums
- CHECKSUM_UNNECESSARY
  - Stack allows two levels (skb->encapsulation)
  - Plan on generalizing (skb->csum_level), will allow up to four

Checksum unnecessary conversion

- Most NICs can provide checksum unnecessary for UDP packets
- If checksum is non-zero, derive checksum complete when processing UDP packet

```
skb->csum = ~pseudo_hdr_csum(skb)
skb->ip_summed = CHECKSUM_COMPLETE
```

- After conversion, any encapsulated checksums is verified by using skb->csum
- Avoid skb_checksum and allow more GRO!

TX checksum offload

- Only one checksum in packet
  - Inner transport checksum (e.g. TCP)
  - NETIF_F_HW_CSUM works
  - NETIF_F_IP_CSUM won’t work unless NIC knows how to parse encapsulation protocol
- Two or more checksums
  - Outer packet (e.g. UDP) and inner transport packet
  - Stack and NICs do not support
  - Alternative: Remote Checksum Offload

Remote Checksum Offload

- Just “like” HW offload except we defer processing to peer
- Useful to provide csum offload for encap’ed packet *and* csum enabled for outer header (e.g. UDP)
- Need 32 bit field in encapsulation header
  - Need extensible encap protocol
  - GUE, geneve :-), … VXLAN, LISP :-(

Remote csum offload operation

- Fields in encap header for encap’ed csum
  - Start - checksum starts relative to UDP header
  - Offset - where to write checksum
- On TX (TCP over UDP example)
  - tcp->check = ~pseudo_inner_hdr_csum
  - encap->start = offset of TCP header
  - encap->check = offset of tcp->check
- On RX
  - Need skb->csum at UDP (either from csum complete or conversion from unnecssary)
  - *(encap->offset) = skb->csum - skb_csum(skb, 0, encap - start)

Minimal HW checksum support

- Combine **checksum unnecessary conversion + remote checksum offload**
- Minimal requirements of NIC are then
  - CHECKSUM_UNNECESSARY for UDP
  - NETIF_IP_CSUM
- This allow csum offload for encapsulation across a large variety of NICs
- Using UDP checksum advantages
  - More coverage over packet, encap hdr (i.e. vnid)
  - Net better performance for encap!

LRO/GRO

- LRO for encap protocols pretty much requires protocol specific deep parsing
- GRO is well supported in Linux
    - Work to make checksum handling consistent with normal path
- *Open question:* should GRO ever to skb_checksum over packet
    - Mixed messages right now
    - GRE allows
    - UDP tunnel doesn’t
    - Can’t configure GRO and no csum offload

TSO/GSO

- Partially generic support
- UDP tunnes
  - SKB_GSO_UDP_TUNNEL
  - SKB_GSO_UDP_TUNNEL_CSUM
  - SKB_GSO_UDP_TUNNEL_RCO ?
- Works with various encaps (e.g. VXLAN, GUE) as long as they don’t have:
  - Seq #’s
  - Packet lengths
  - Checksum/packet authentication
  - Anything that must be uniquely set per segment

TSO/LRO to guest driver

- Greatest value in segmentaiton offloads when plumbing then guest OS <-> host driver (device)
- On TX, guest uses TSO interface, host kernel converts to TSO/GSO
- On RX, host probably uses GRO, converts to LRO to guest device

Other rx offloads

- Protocol specific hash, LRO, packet steering
- rx-filter: Destination UDP port->action, queue
- Example: deep parsing for a flow hash
  - UDP from Internet may not be able to arbitrarily set source port (e.g. to go though stateful NAT)
  - Deep parsing in this case may have merit
- Should not affect core stack
  - Stack should not care that UDP sockets are tunnels or what protocol is running over them
  - IMO, ndo_add_vxlan_port not necessary (should not be extended for other encap protocols)