Deal (efficently?) with multiple UDP encapsulation layers

Double UDP encap gets no love from the stack

- All UDP tunnel devices expose 0 hw_enc_features and 0 gso_partial_features:
  - On TX side GSO over UDP tunnel are segmented at dev_queue_xmit() time on such devices
- GRO explicitly stops after the first UDP encap layer (skb->encapsulation and NAPI_GRO_CB(skb)->encap_mark checks)
- When inner encap usually lives in a different namespace (or in a VM) gro-cells on the outer encap device still can't aggregate incoming packets.

Transmit path and GSO partial

- UDP tunnel devices could expose UDP_TUNNEL* as GSO partial features
  - Need to review inner header updating - probably fine: currently the inner headers are reset/initializated at dev_queue_xmit() time over UDP tunnels by iptunnel_handle_offloads(),
  - Without the GRO counterpart will make TX and RX very asymmetric

Restricting the problem

- Geneve has a rather large option space (252 bytes), it could fit one (or more...) encapsulation level (70 bytes overall)
- options support inside the is currently ~NULL, *except* at GRO time; GRO properly match options contents before aggregation

Rough implementation details

- Conditionally enabled via NL option, default off
- Geneve devices expose full GSO feature set in hw_enc_features, including UDP_TUNNEL*
- When handling encapsulated packets, geneve_xmit_skb():
  - strips the outer headers from the packets before adding the geneve header
  - when adding the header includes a new/custom option. The option contents are the previously stripped headers, verbatim

Rough implementation details [II]

- On the wire and at GRO time the packet has a single UDP encapsulation level - with an 'unusually large' geneve header, the GRO engines will not need any change.
- When processing an incoming packet, genve_rx() looks for the new/custom option, if found push into the packet the headers carried by the option. It need to adjust:
  - The skb (outer) hdr fields
  - The header csum and len related fields - the values carried on the wire should be ignored.

Even more implementation details

- Stripping the headers at xmit time and pushing them at RX time does not require actual memcpy(), just move the skb offsets
- We probably want to include into the option even some additional metadata with the carried headers lengths (and carefully validate them on reception)
- The mechanism allows for several UDP encap layers before exhausting the geneve option space.

Problems

- Endpoints not supporting the custom option become blackholes - will have the 'critical' bit set, causing old kernel to drop them. How to detect such blackholes?
  - We can't hook into the TCP blackhole detection, as the TCP endpoint could be in a different netns (or a in VM)
  - Geneve-level validation blackhole detection via more options?

More problems

- For GSO packets, the headers carried by the geneve option have invalid length/csum fields on the wire (pre-segmentation values), should we just zero them?
- Is TSO impacted? No clear idea looking at the driver code
- Is geneve H/W offload (RSS) impacted? (same as above)
- Would nesting support for single UDP encap protocol be enough/worthy? (yes :)
    - vxlan- GPE allows for a similar mechanism using custom shim protocols

Blackhole detection via sibling echo option

- Each geneve (lwt) device with nested tunnel enabled runs a periodic probe timer
- At timer expiration time sends towards the destination a geneve-level ‘echo’ using a newly defined critical echo option and empty payload
- A complying peer, upon ‘echo’ reception must reply with a similar ‘echo-replay’ geneve packet
- Reception of an echo or echo reply packets enable to tunnel to generate ‘nested tunnel option’ for ~ probe time