The Sunset of VXLAN

The Rise of Geneve

State of VXLAN

VXLAN has been around for almost a decade, plenty of deployments, NANOG 74 content is full of VXLAN. What might be wrong?

```
nanog74 > grep -i VXLAN -ho 201810*.txt | wc -l
63

nanog74 > grep -i VXLAN -ho 201810*.txt | sort | uniq -c
      2 vxLAN
     49 VxLAN
     12 VXLAN
```

Just please `s/vxLAN/VXLAN/gI` pretty please!

History of VXLAN

VXLAN was a successful accident.

- Designed by a single vendor for a single product family.
- Designed for a specific and bounded use case.
- Not a product of IETF WG, it was an individual submission.
- The engineering tradeoffs were known and accepted.

- VXLAN got successfully used and abused for things not even envisioned at the design time, and the limitations are now evident.

The problems of VXLAN

Several large problem areas are present in VXLAN as an encapsulator:

- No protocol identifier.
- No indicator of non-client payload.
- No extensibility.

Protocol Identifier

- There is no payload type identifier.
- A single tunnel cannot carry more than one payload type.
- VNI namespace is large, but that is not a practical scaling problem.
- Number of supported tunnels is a far bigger practical scalability problem.
- Originally VXLAN was envisioned for carrying Ethernet payload only.

Non-client Payload

- Everything in the tunnel is a payload. A client payload.
- This rules out majority of OAM mechanisms.
- Running traditional OAM toolkits over VXLAN will provide you some data. The quality of that data is under question.
- Large portion of OAM toolkits is hardly compatible with VXLAN (eg, BFD).
- Client cooperation is required and assumed.
- OAM mechanisms cannot just be added on top of some network protocol – that needs to be architected in from the start.

No Extensibility

- All fields in VXLAN header have predefined values.
- While only a few are actually used.
- No possibility to add extensions in an interoperable manner.
- There are proprietary VXLAN extensions.

Requirements for a new encapsulator

- IETF NVO3 WG

- Extensibility.
- HW friendliness (TLVs vs bit fields).
- Middleboxes.
- Security aspects.
- Practical implementation aspects (software is easy, hardware is in fact hard).

Evolution of VXLAN

- Geneve – a compromise between functionality and HW implementation complexity.
- GUE – perceived to be too complex to implement.
- VXLAN-GPE – perceived to be not extensible enough.
- Some niche proposals, did not advance further.

- Geneve is the proposed successor to VXLAN.

Geneve

- TLV based extension headers.
- HW friendly vendor extensibility mechanism.
- Header integrity.
- Possible payload encryption
- Payload type indicator.
- OAM indicator.
- Base encapsulator header is 8 octets, up to 260 octets for extensions total.

- Works like VXLAN, just better.

Design aspects

- VXLAN is starting to show its age.
- Be careful with new designs.
- Especially if OAM or interoperability is needed.
- Component vendors are ready.
- System vendors are getting there.
- Architects and operators need to be aware.

- The changes are in the data plane. Control plane components stay the same.