HELIUM

Hybrid Encapsulation Layer for IP and UDP Messages

Ben Schwartz

IETF 102, Dispatch

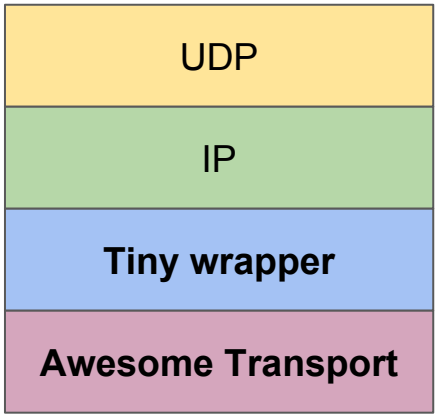
Goal: UDP Proxying for HTTP/QUIC and Beyond

- HTTP is self-proxying: `GET http://other.domain.example/foo HTTP/1.1`
- HTTPS is too: `CONNECT other.domain.example:443 HTTP/1.1`
- What about HTTP/QUIC?
- What about …
  - WebRTC (currently TURN)
  - VPNs (like OpenConnect, OpenVPN, L2TP)
  - In-betweeny things (e.g. UDP + ICMP)
- Can we find a protocol that
  - supports all these use cases
  - is simple to define
  - can run on top of HTTP
  - doesn’t require HTTP
  - enables good performance

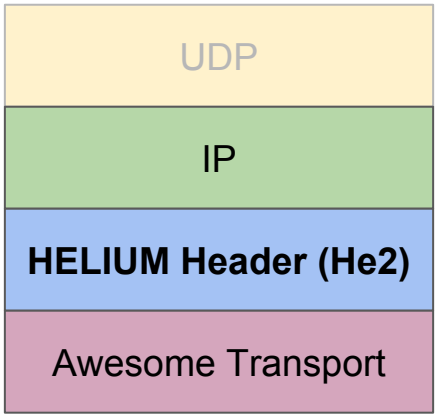
What’s a UDP proxy really (e.g. TURN)?

- UDP payload
- Outbound packets
  - Destination IP
  - Destination UDP port
  - DONT-FRAGMENT*
- Inbound packets
  - Source IP
  - Source UDP port
- Stable port mapping
  - Bound port to tell peer

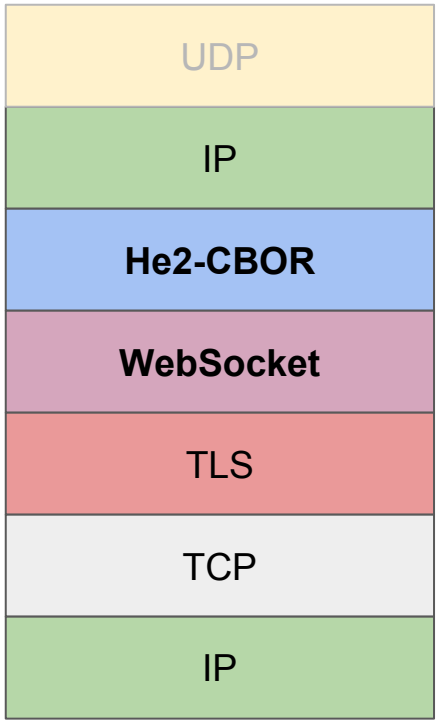
HELIUM Protocol Stack



Idea



HELIUM (abstract)



HELIUM-WebSocket

HELIUM Header in a nutshell: 3 msg types

- Sending a packet: **outbound**
  - optional number **id**: to request a “meta” reply
  - optional string **domain**: to override the destination address with a **DNS name**
  - optional number **dns**: to override the destination address with a **DNS server**
  - **packet!**
- Receiving a packet: **inbound**
  - uint32 **timestamp**: when the packet was received (microseconds)
  - **packet!**
- Finding out what happened to your packet: **meta**
  - number **id**: the outbound packet id
  - optional integer[] **errors**: any error codes that prevented the packet from being sent
  - uint32 **timestamp**: when the packet was sent (microseconds)
  - **packet prefix** including any modified portions of the outbound packet

IP as a proxy protocol, ICMP-style

`If the proxy modified the outbound packet in any way, the "meta" message MUST contain a` **`prefix of the outbound packet as sent, including any parts that were modified.`** `Changes might include the source IP, destination IP, TTL, DSCP priority, UDP source port, etc.`

- Inspired by ICMP error responses
- Reuse IP as the client-proxy protocol: **no need to invent a new one**
- No artificial limitations: try to send whatever you want and see how the proxy mangled it.
  - Not limited to UDP! Can do UDP + ICMP (PMTUD! Traceroute!) or even a full VPN.
  - Can potentially proxy TTL, ECN, DSCP, Jumbograms, fragments, etc.

Other tricksy features

- UDP + ICMP mode can be implemented without root
- Microsecond timestamps for delay-based congestion control
- Domain override: minimize latency for named destinations
- DNS server index: send advanced queries to the proxy’s recursives
- Proxy can offload fragment reassembly to the client
- Bind an address by sending to 0.0.0.0 and inspecting the prefix in the reply

HELIUM-WebSocket Proxy Discovery

CONNECT foo.example:443 HTTP/1.1
Host: proxy.example
Proxy-Authorization: basic YWxhZGRpbjpvc
…

HTTP/1.1 200 OK
Helium-Proxy-URL: wss://proxy.example/foo

GET /foo HTTP/1.1
Host: proxy.example:443
Upgrade: websocket
Connection: Upgrade
Proxy-Authorization: basic YWxhZGRpbjpvc
Sec-WebSocket-Protocol: helium-cbor
Sec-WebSocket-Extensions: permessage-deflate
…

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Protocol: helium-cbor
Sec-WebSocket-Extensions: permessage-deflate

Alternative taglines

- “So that’s what the NAT did”
- “A proxy is an honest middlebox”
- “ICMP for the JSON era”