# Transport-split Proxying for encrypted traffic

# Performance Enhancing Proxies

- RFC 3135 "Performance Enhancing Proxies intended to mitigate link layer degradation"

- RFC 3449 "TCP Performance Implications of Network Path Asymmetry"

# Problems PEPs solve

- Improve performance of the transport when crossing low-performance links (high-loss, high-latency) or highly asymmetric links (narrow upstream bandwidth)
    - Examples include but are not limited to: satellite, mobile network, wireless
- Put a PEP on one (sometimes both) end(s) of the low-quality / asymmetric link and perform various kinds of magic, including:
    - ACK manipulation (suppression, reconstruction, compaction)
    - Receive window size manipulation
    - Header compression

# Problems PEPs create

- Typical middlebox, so the usual caveat applies:

> [RFC 6182] *All these middleboxes optimize current applications at the expense of future applications. In effect, future applications will often need to behave in a similar fashion to existing ones, in order to increase the chances of successful deployment. Further, the precise behavior of all these middleboxes is not clearly specified, and implementation errors make matters worse, raising the bar for the deployment of new technologies*

- Nasty when it completely breaks, but even nastier when it only *partially* breaks the end-to-end path: e.g., when it ends up "eating" unknown (to the box) TCP options

# PEPs and encrypted traffic

- Transport header protection (e.g., QUIC, IPsec ESP / AH)
  - Transport header can't be modified / forged, therefore PEP functions are completely inhibited:
    - ☑ NO header compression
    - ☑ NO ACK tricks
    - ☑ NO rwin tricks

# QUIC, specifically

- ACK is inside the encrypted envelop

- PN visible (as of draft-ietf-quic-transport-10), but not for long (PR #1079)

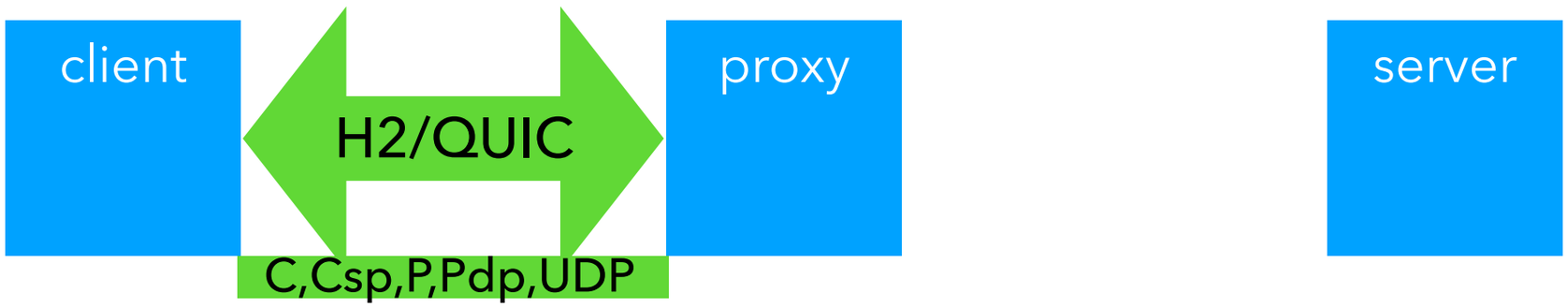
# Questions

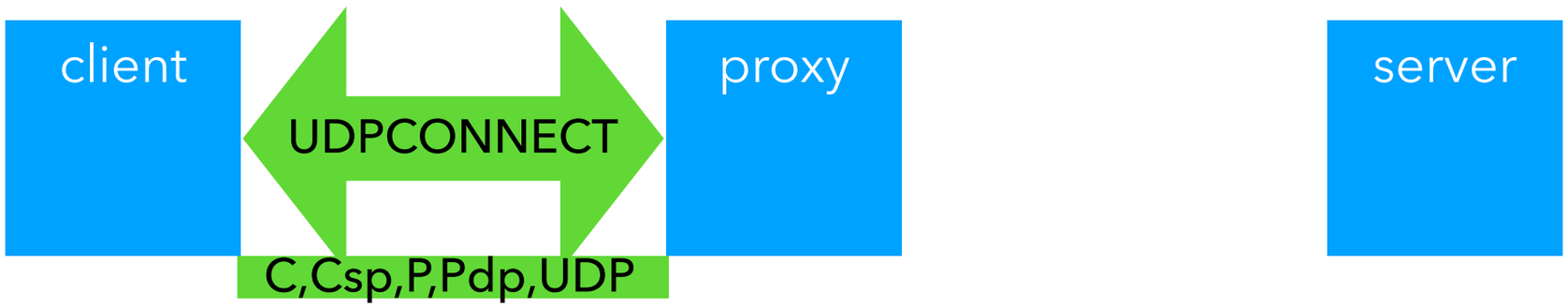
- Is PEP still a valid approach?
    - Are the problems it solves still relevant problems?
        - Can the problems it solves be addressed with other techniques?
- If PEPs are an unavoidable necessity, what can be done WRT QUIC?

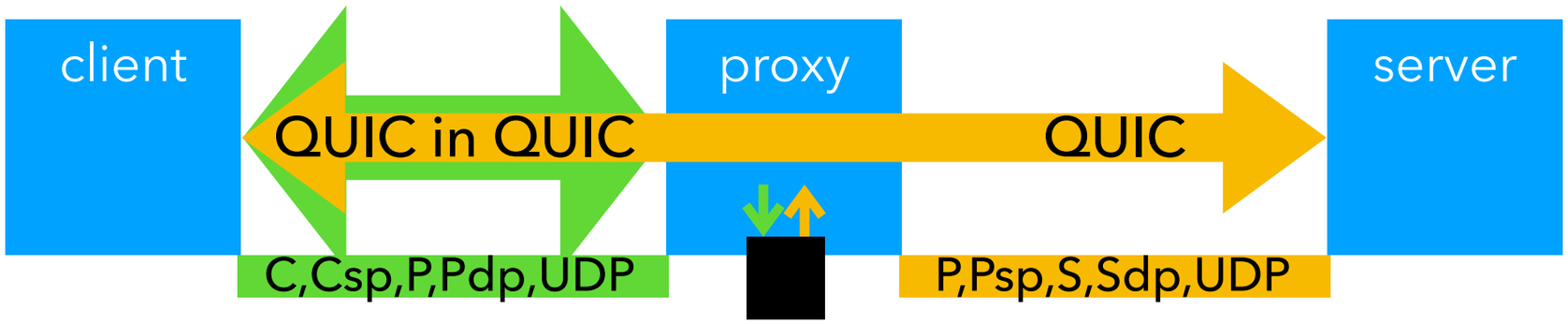
# UDPCONNECT to the rescue?

- New HTTP method currently under design

- Semantically equivalent to HTTP CONNECT, but used for building UDP tunnels instead of TCP


client
proxy
server
H2/QUIC
C,Csp,P,Pdp,UDP


client
proxy
server
UDPCONNECT
C,Csp,P,Pdp,UDP


client
proxy
server
QUIC in QUIC
QUIC
C,Csp,P,Pdp,UDP
P,Psp,S,Sdp,UDP

# What can we do with it?

- UDPCONNECT is the low-hanging fruit - we could have it standardised probably without too much friction
  - We are thinking about more radical transformations of QUIC's underlying security association model, but these have much lower chance of adoption in the short term

- Is UDPCONNECT enough to do at least *some* kind of transport optimisation?