On-Path Proxy Discovery

Side meeting at IETF-121, Dublin
Thursday, 7 November, 18:30-20:00, Wicklow Hall 2A

All material: https://github.com/mwelzl/oppd

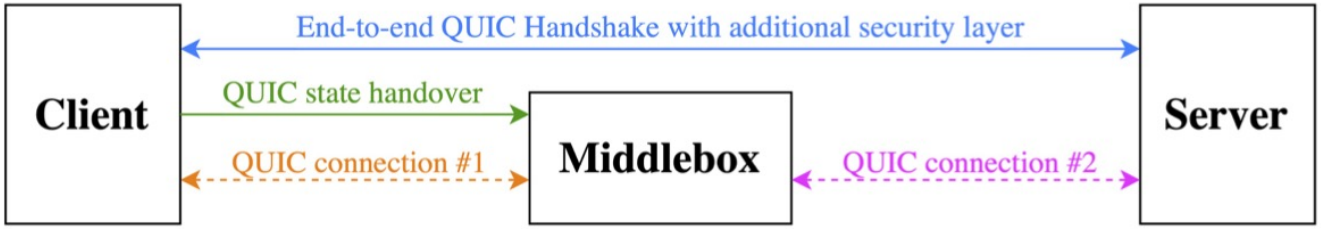
Internet-draft: draft-welzl-panrg-oppd-00

Why OPPD?

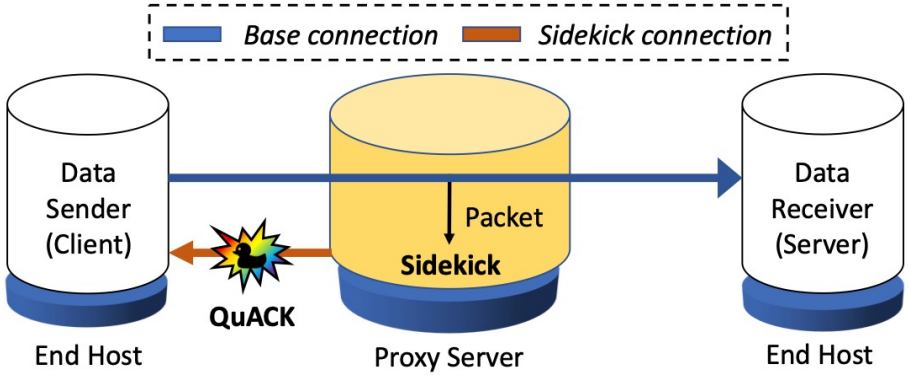
- Performance Enhancing Proxies (PEPs) did some useful things for TCP
  - E.g., improved communication over satellite
  - And they caused problems, because they were transparent
- QUIC requires proxies to be non-transparent
  - So now, we can get them right: let the endpoint choose
  - This is good in general, also for TCP => but we talk about QUIC
- Transparent proxies are inherently on-path
  - For QUIC, we *have* to discover them

Two use cases: both need OPPD

- Example 1: "Secure Middlebox-Assisted QUIC" (**SMAQ**)
  https://ieeexplore.ieee.org/document/10186363 Preprint: https://arxiv.org/abs/2307.08543



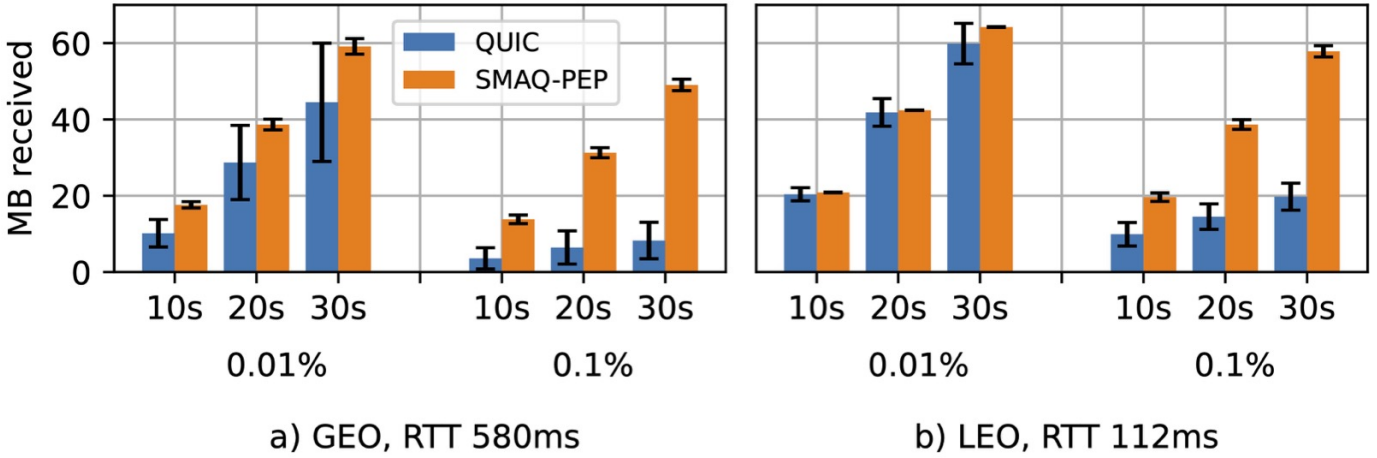
- Example 2: "**Sidekick**: In-Network Assistance for Secure End-to-End Transport Protocols" https://www.usenix.org/conference/nsdi24/presentation/yuan



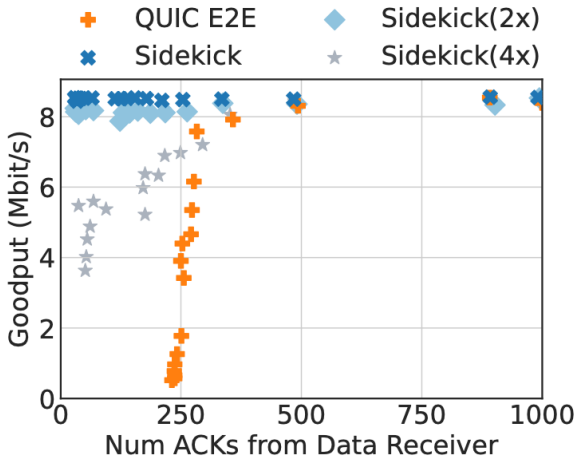
There can be more, of course: we want a generic solution.

Is this really worthwhile?

- SMAQ satellite tests show significant improvements



- Sidekick: paper evaluated 3 use cases: earlier retransmission of packets lost due to noise, conn-splitting, ACK reduction



Reducing WiFi receiver ACKs for lower collisions. High goodput independent of e2e ACK frequency.
10 Mb upload

Why the "on-path" limitation?

- Makes use of default routing
  - No need to involve the operator (e.g., proxy could be a WiFi AP)
  - Can be very lightweight

- No penalty if there is nothing on the path (common case)
  - Should therefore be faster than other discovery schemes

Requirements

- Needs to work through NATs, i.e. use same 5-tuple as base connection (the connection to be optimized)

- Endpoints announce their interest
  - Else, potentially many unsolicited packets, and unclear for a proxy when to send one

- Proxies answer in a way that proves that they've seen packets from the endpoint (think ICMP for a simple case)

- The next step is out of scope for now
  - E.g., first packet from proxy could contain IP address + port number to use from now on

How could this work?

draft-welzl-panrg-oppd
identifies 3 signaling options

(assuming: the same in both directions,
for now, for simplicity)

1. a "special" packet

- If a "special" packet from the sender arrives at the receiver, there, QUIC's crypto code will recognize it as an error and ignore it.

- Pros:
  - Only change e.g. QUIC implementation + proxy
- Cons:
  - Extra packets
  - A quite dirty hack
  - Proxy needs to spoof source address / port

2. TCP / UDP options

- Pros:
  - No spoofing
  - No extra packets
  - Endpoint: ignored in the OS if not supported
- Cons:
  - Changes to both QUIC and the OS
  - Can it even be done? UDP options must not be changed in transit
  - MTU problems?

3. QUIC transport option

- In the beginning; decipherable by anybody because there's no established secret yet.

- Pros:
  - No extra packets
  - In line with part of QUIC spec
- Cons:
  - Specific to QUIC only (design per transport)
  - Heavier work for the proxy
  - Not suitable as a channel back from proxy in the general case (works for SMAQ)

Other considerations

- Multiple proxies on the path
  - They can all announce their existence ("options" signaling: add themselves to a field, provided there's enough space, or insert a new option...)

- Multipath
  - Nothing special to do here? This is per sub-path discovery.
  - Endpoints initiate OPPD for every sub-path where they want to use a proxy in case it's available.

What do we want from this?

1. Where should we do this work?

2. Where should we do following work on performance enhancing functions? (SMAQ / Sidekick / .. ?)

3. If time permits: design discussion of OPPD