DNS-over-QUIC
(DoQ)
draft-ietf-dprive-dnsoquic

Christain Huitema
Sara Dickinson
Allison Mankin
DoQ - Background

- **April 2017** - First Draft in QUIC WG (stub to resolver ONLY)
  - Since then - Christian has worked on QUIC and DoQ implementations

- **March 2019** - IETF 104 some hackathon work….

- **March 2020** - Draft moved to DPRIVE WG (adopted Apr 2020)
DoQ - Background

● **Nov 2020** - (IETF 109) DPRIVE WG meeting
  ○ Still no clear answer on whether WG wants to pursue for stub-rec
     ■ Is it needed? Is it (more) performant?
  ○ Asked to consider extending draft to cover rec-auth and XFR

● **Dec 2020** - AgGuard launched first DoQ resolver service

● **Feb 2021** - Core QUIC docs now in RFC Editor Queue
● **Feb 2021** - XoT draft passes WGLC
Updates in -02 draft (Feb 2020)

- Add implementation section
- Add appendix on how to potentially support XFR
- Update IANA Considerations: Plan to request port 8853
  - Port 784 used previously for experiments
- Minor updates related to transport parameters
  - Based on feedback from interop between picoquic and aioquic. Thanks, Stéphane!
# Implementation status (all open source)

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Language</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreDNS</td>
<td>Go</td>
<td>AdGuard use as DoQ server</td>
</tr>
<tr>
<td>AdGuard DNS Proxy</td>
<td>Go</td>
<td>Simple proxy or server supporting DoQ (used in ADGuard Home)</td>
</tr>
<tr>
<td>dnslookup</td>
<td>Go</td>
<td>Command line utility wrapper for Adguard DNS proxy</td>
</tr>
<tr>
<td>AdGuard C++ DNS libs</td>
<td>C++</td>
<td>AdGuard use in mobile app</td>
</tr>
<tr>
<td>Quicdoc</td>
<td>C</td>
<td>Simple DoQ impl based on Picoquic</td>
</tr>
<tr>
<td>aioquic</td>
<td>Python</td>
<td>QUIC implementation includes example DoQ client/server</td>
</tr>
<tr>
<td>Flamethrower</td>
<td>C++</td>
<td>DNS performance utility with experimental DoQ</td>
</tr>
</tbody>
</table>
Implementation status (all open source)

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Language</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoreDNS</td>
<td>Go</td>
<td>AdGuard use as DoQ server</td>
</tr>
<tr>
<td>AdGuard DNS Proxy</td>
<td>Go</td>
<td>Simple proxy or server supporting DoQ (used in AdGuard Home)</td>
</tr>
<tr>
<td>dnslookup</td>
<td>Go</td>
<td>Command line utility wrapper for Adguard DNS proxy</td>
</tr>
<tr>
<td>AdGuard C++ DNS libs</td>
<td>C++</td>
<td>AdGuard use in mobile app</td>
</tr>
<tr>
<td>Quicdoc</td>
<td>C</td>
<td>Simple DoQ impl based on Picoquic</td>
</tr>
<tr>
<td>aioquic</td>
<td>Python</td>
<td>QUIC implementation includes example DoQ client/server</td>
</tr>
<tr>
<td>Flamethrower</td>
<td>C++</td>
<td>DNS performance utility with experimental DoQ</td>
</tr>
</tbody>
</table>

- **Performance Measurements**
  - None yet AFAWK
  - But... AdGuard claim performs well in mobile environment (lower bandwidth, handles packet loss better, connection migration)
How to support Recursive to Authoritative?

- No real practical difference to DoT for discovery
  - Both have dedicated port and same authentication model
How to support Recursive to Authoritative?

- No real practical difference to DoT for discovery
  - Both have dedicated port and same authentication model

- Issue arises for XFR support....
  - Current draft uses very simple ‘clean’ mapping:
    - **1 DNS query/response pair maps to 1 QUIC stream**
      (stream is closed by each end immediately after sending)
    - StreamData is a UDP-like payload (Query ID=0) - no 2 byte length field

  - But...XFRs can have multiple responses....

---

<table>
<thead>
<tr>
<th>Single QUIC connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>STREAM 4: Query</td>
</tr>
<tr>
<td>STREAM 8: Query</td>
</tr>
<tr>
<td>STREAM 8: Response</td>
</tr>
<tr>
<td>STREAM 4: Response</td>
</tr>
<tr>
<td>STREAM 12: Query</td>
</tr>
</tbody>
</table>
How to support XFR?

- **Initial proposal** in Appendix of current draft describes an alternative mapping
  - Prepend ALL DNS messages with a 2 byte length field
  - Relax mapping to allow multiple responses on a single stream for an XFR query
    (In practice, DoQ stream content is similar to a TCP connection)

- **Pros**: Supports XFR, ALPN allows backwards compatibility
- **Cons**: Complicates mapping, small overhead, new error conditions
How to support XFR?

- **Initial proposal** in Appendix of current draft describes an alternative mapping
  - Prepend ALL DNS messages with a 2 byte length field
  - Relax mapping to allow multiple responses on a single stream for an XFR query
    (In practice, DoQ stream content is similar to a TCP connection)

- **Pros**: Supports XFR, ALPN allows backwards compatibility
- **Cons**: Complicates mapping, small overhead, new error conditions

- Range of other possibilities to also consider…
  - Require server to add 2 byte length field ONLY to XFR responses
  - Use separate server initiated streams for each XFR response
  - Define new stream type for XFRs within DoQ
  - Define XFR-over-QUIC as a separate protocol with different ALPN
    - NOTE: QUIC supports server initiated streams so a PUSH model is possible
Questions for Working Group

- Implementation and operational progress
  - Why not move forward with specification?
  - Is more performance data still needed?

- Scope: stub-rec, rec-auth or both?

- If rec-auth is included, how/if to handle XFR support?
Backup slides
## What is DoQ?

<table>
<thead>
<tr>
<th>DNS over QUIC (DoQ)</th>
<th>QUIC</th>
<th>UDP</th>
<th>IP</th>
</tr>
</thead>
</table>

- **Simple mapping of DNS over dedicated QUIC connections**
  - One QUIC Stream per DNS Query/Response
  - Query and Response size up to 64K (65536)
  - Parallel processing, no head of queue blocking
  - QUIC handles timers, retransmissions, connection management

- **Draft currently targets the stub-recursive scenario**
  - Recursive-authoritative requires discovery

- **Operates on dedicated port (TBC-IANA)**
Example flow, First connection, 1-RTT

Initial packet (Client Hello)

Initial (Server Hello) + Handshake (extensions, certificate, server finished)

Handshake (ACK, Client finished) + 1-RTT (Stream 0: DNS query)

1-RTT (Handshake Done, Session Ticket, ACK, Stream 0: DNS Response)

1-RTT (ACK)

- QUIC handshake embeds TLS handshake
  - Size of server responses depends on size of server certificate, signature

- DNS Query can be sent as soon as server first flight is received

- Response arrives after 2-RTT plus service time.
Example flow, Second connection, 0-RTT

- Session Ticket obtained during previous connection
- DNS Query sent immediately as 0-RTT data
- DNS Response sent with first server flight
- Response arrives after 1-RTT plus service time.
Example flow, DNS Queries (following HS)

- Each Query uses a new QUIC stream (Query-ID is always 0)
- Responses can arrive in any order
DoQ vs DoT vs DoH3?

- Differences with DoT
  - QUIC instead of TLS + TCP

- Difference with DoH3
  - DoH3 has integration with the Web
  - DoQ does not need to use the HTTP-3 layer
  - DoQ has no dependency on HTTP platforms

- With ESNI/ECHO, all 3 solutions can cross firewalls