DNS-over-QUIC
First experience with DoQ

Andrey Meshkov
CTO and Co-Founder of AdGuard
am@adguard.com
@ay_meshkov
DNS-based products by AdGuard

- AdGuard DNS — public DNS resolver
- AdGuard Home — DNS server for personal use with content blocking capabilities
- AdGuard apps provide DNS filtering and encryption capabilities (DoH/DoT/DNSCrypt)
- Recently we’ve added DoQ support to all of them: [https://adguard.com/en/blog/dns-over-quic.html](https://adguard.com/en/blog/dns-over-quic.html)
AdGuard DNS

- Public DNS resolver with the focus on content blocking
- The first beta was launched in the end of 2016
- Officially released in December, 2018
- Open-source
  https://github.com/AdguardTeam/AdGuardDNS
- Most of the clients are mobile devices
AdGuard DNS

Avg 100,000+ RPS

DNS: 46%
DoT: 37%
DoH: 15%
DNSCrypt: 1%
DoQ: 1%
DNS Encryption

Different protocols pros and cons

- Plain DNS - fast, no encryption
- DNSCrypt - fast, non-standard encryption
- DoT - slow, standard encryption
- DoH - slow but practical, standard encryption, more data points that can be potentially used for fingerprinting
- DoT/DoH bandwidth is x2.5 compared to DNS
QUIC vs TCP+TLS

- Faster handshake
- Solves head-of-line blocking
- Connection migration
Faster Handshake

HTTP Request Over TCP + TLS

Client

TCP SYN
TCP SYN + ACK
TCP ACK
TLS ClientHello
TLS ServerHello
TLS Finished
HTTP Request
HTTP Response

Server

HTTP Request Over QUIC

Client

QUIC
QUIC
QUIC
HTTP Request
HTTP Response

Server

Images from https://blog.cloudflare.com/the-road-to-quic/
HTTP/2 head-of-line blocking: a single TCP packet loss will, all queries/responses have to wait

QUIC - every DNS query/response is a new QUIC stream
Connection Migration

- Endpoints can use “Connection ID” to track connections
- This makes it possible to continue using the same connection when network change occur (i.e. Wi-Fi <-> Cellular)

QUIC packet header

<table>
<thead>
<tr>
<th>Public Flags (8)</th>
<th>Connection ID (0, 8, 32 or 64)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUIC Version (32) (optional)</td>
<td>Packet Number (8, 16, 32 or 48)</td>
</tr>
</tbody>
</table>
**DoQ vs DNS-over-HTTP/3**

- Both DoQ and DoH3 use QUIC as an underlying transport
- HTTP/3 adds HTTP on top of it
- HTTP adds almost zero value
- It adds more data-points that can be used for fingerprinting clients

**Examples:**
- HTTP headers order
- TLS properties
- ETag tracking
DoQ Server-Side Implementations

- CoreDNS fork:
  https://github.com/AdguardTeam/coredns

```
1 quic://.:784 {
2   tls certs/example.crt certs/example.key
3   forward 94.140.14.14
4 }
```

Sample CoreDNS configuration
DoQ Server-Side Implementations

- dnsproxy:
  [https://github.com/AdguardTeam/dnsproxy](https://github.com/AdguardTeam/dnsproxy)

```
./dnsproxy \
  -l 127.0.0.1 \
  --quic-port=784 \
  --tls-crt=example.crt \
  --tls-key=example.key \
  -u 8.8.8.8:53 \n  -p 0
```

*Running dnsproxy as a DoQ server forwarding queries to 8.8.8.8*
DoQ Server-Side Implementations

- AdGuard Home:
  https://github.com/AdguardTeam/AdGuardHome

DNS-over-QUIC port (experimental)

784

If this port is configured, AdGuard Home will run a DNS-over-QUIC server on this port. It's experimental and may not be reliable. Also, there are not too many clients that support it at the moment.
DoQ Client-Side Implementations

- dnsproxy (written in Golang, can be used as a library): https://github.com/AdguardTeam/dnsproxy
- AdGuard Home (written in Golang, uses dnsproxy internally): https://github.com/AdguardTeam/AdGuardHome
- DnsLibs (library, written in C++): https://github.com/AdguardTeam/DnsLibs
QUIC Implementations

- **Golang:** quic-go
  
  [https://github.com/lucas-clemente/quic-go](https://github.com/lucas-clemente/quic-go)
  
  *Does not support connection migration yet.*

- **C++:** ngtcp2
  
  [https://github.com/ngtcp2/ngtcp2](https://github.com/ngtcp2/ngtcp2)
Current issues

- Connection migration is not supported by AdGuard DNS:
  - Not yet implemented in `quic-go`
  - We use ECMP to balance load between servers in the same location which is not compatible with connection migration

- QUIC and DoQ are still drafts:
  - They’re not likely to change much, though
Feedback

• Users’ feedback ranges from very positive to neutral

  vikingr666 5 points · 25 days ago

  Seems to be working pretty well for me on iOS!

• We’re yet to get the precise numbers, but for now it seems that:
  ○ The share of networks where DoQ is blocked, is quite small
  ○ It does provide advantage over DoH in cellular data networks, as expected
Thank you!

Questions?

Andrey Meshkov
am@adguard.com
@ay_meshkov