RFC 9250 : DNS over Dedicated QUIC Connections

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 mai 2022

Date de publication du RFC : Mai 2022

https://www.bortzmeyer.org/9250.html

Ce nouveau RFC complète la série des mécanismes de protection cryptographique du DNS. Après DoT (RFC 7858 [1]) et DoH (RFC 8484), voici **DoQ**, DNS sur QUIC <https://www.bortzmeyer.org/quic.html>. On notera que bien que QUIC ait été conçu essentiellement pour les besoins de HTTP, c'est le DNS qui a été la première application normalisée de QUIC.

Fonctionnellement, DoQ est très proche de ses prédécesseurs, et offre les mêmes services de sécurité, confidentialité, grâce au chiffrement, et authentification du serveur, via le certificat de celui-ci. L'utilisation du transport QUIC <https://www.bortzmeyer.org/quic.html> permet quelques améliorations, notamment dans le « vrai » parallélisme. Contrairement à DoT (RFC 7858) et DoH (RFC 8484), DoQ peut être utilisé pour parler à un serveur faisant autorité <https://www.bortzmeyer.org/serveur-dns-faisant-auto
html> aussi bien qu'à un résolveur <https://www.bortzmeyer.org/resolveur-dns.html>.

Notez que le terme de DoQ ("*DNS over QUIC*") n'apparait pas dans le RFC de terminologie DNS, le RFC 8499. Il sera normalement dans son successeur.

Logiquement, DoQ devrait faire face aux mêmes problèmes politiques que ceux vécus par DoH <https://www.bortzmeyer.org/doh-et-ses-adversaires.html> (voir aussi mon article dans Terminal <https://journals.openedition.org/terminal/8221>) mais cela n'a pas été encore le cas. Il faut dire qu'il y a peu de déploiements (ne comptez pas installer DoQ comme serveur ou client tout de suite, le code est loin d'être disponible partout).

Le cahier des charges de DoQ est (sections 1 et 3 du RFC) :
— Protection contre les attaques décrites dans le RFC 9076.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7858.txt

— Même niveau de protection que DoT (RFC 7858, avec notamment les capacités d'authentification variées du RFC 8310). Rappelez-vous que QUIC est **toujours** chiffré (actuellement avec TLS, cf. RFC 9001).
— Meilleure validation de l'adresse IP source qu'avec le classique DNS sur UDP, grâce à QUIC.
— Pas de limite de taille des réponses, quelle que soit la MTU du chemin.
— Diminution de la latence <https://www.bortzmeyer.org/latence.html>, toujours grâce à QUIC et notamment ses possibilités de connexion 0-RTT (RFC 9000, section 5), ses procédures de récupération en cas de perte de paquets (RFC 9002) et la réduction du "*head-of-line blocking*", chaque couple requête/réponse étant dans son propre ruisseau.

En revanche, DoQ n'esssaie **pas** de :

— Contourner le blocage de QUIC que font certaines "*middleboxes*" (DoH est meilleur, de ce point de vue). La section 3.3 du RFC détaille cette limitation de DoQ : il peut être filtré trop facilement, son trafic se distinguant nettement d'autres trafics.
— Ou de permettre des messages du serveur non sollicités (pour cela, il faut utiliser le RFC 8490).

DoQ utilise QUIC <https://www.bortzmeyer.org/quic.html>. QUIC est un protocole de transport généraliste, un concurrent de TCP. Plusieurs protocoles applicatifs peuvent utiliser QUIC et, pour chacun de ces protocoles, cela implique de définir comment sont utilisés les ruisseaux ("*streams*") de QUIC. Pour HTTP/3, cela a été fait dans le RFC 9113, publié plus tard. Ironiquement, le DNS est donc le premier protocole à être normalisé pour une utilisation sur QUIC. Notez qu'une autre façon de faire du DNS sur QUIC est de passer par DoH et HTTP/3 (c'est parfois appelé DoH3, terme trompeur) mais cette façon n'est pas couverte dans notre RFC, qui se concentre sur DoQ, du DNS directement sur QUIC, **sans** HTTP.

La spécification de DoQ se trouve en section 4. Elle est plutôt simple, DoQ étant une application assez directe de QUIC. Comme DoQ utilise QUIC, il y a forcément un ALPN, en l'occurrence via l'identificateur `doq`.

Le port, quant à lui, est par défaut 853 <https://www.iana.org/assignments/service-names-port-num
service-names-port-numbers.xml>, également nommé `domain-s`. C'est donc le même que DoT, et aussi le même que le DNS sur DTLS (RFC 8094, protocole expérimental, jamais vraiment mis en œuvre et encore moins déployé, et qui est de toute façon distinguable du trafic QUIC, cf. section 17.2 du RFC 9000). Bien sûr, un client et un serveur DNS majeurs et vaccinés peuvent toujours se mettre d'accord sur un autre port (mais pas 53, réservé au DNS classique). Cette utilisation d'un port spécifique à DoQ est un des points qui le rend vulnérable au filtrage, comme vu plus haut. Utiliser 443 peut aider. (Le point a été chaudement discuté à l'IETF, entre défenseurs de la vie privée et gestionnaires de réseau qui voulaient pouvoir filtrer facilement les protocoles de leur choix. Si on utilise le port 443, il faut se rappeler que l'ALPN est pour l'instant en clair, la lutte de l'épée et de la cuirasse va donc continuer.)

Plus important, la question de la correspondance entre les messages DNS et les ruisseaux QUIC. La règle est simple : pour chaque requête DNS, un ruisseau est créé. La première requête sur une connexion donnée sera donc sur le ruisseau 0 (RFC 9000, section 2.1), la deuxième sur le ruisseau 4, etc. Si les connexions QUIC sont potentiellement de longue durée, en revanche, les ruisseaux ne servent qu'une fois. Le démultiplexage des réponses est assuré par QUIC, pas par le DNS (et l'identificateur de requête DNS est donc obligatoirement à zéro, puisqu'inutile). Le parallélisme est également fourni par QUIC, client et serveur n'ont rien de particulier à faire. Il est donc recommandé de ne pas attendre une réponse pour envoyer les autres questions. Les messages DNS sont précédés de deux octets indiquant leur longueur, comme avec TCP.

Comme avec TCP, client et serveur ont le droit de garder un œil sur les ruisseaux et les connexions inactifs, et de les fermer d'autorité. La section 5.5 donne des détails sur cette gestion de connexion, mais le RFC 7766 est également une bonne lecture (voir aussi le RFC 9103 pour le cas des transferts de zones).

DoQ introduit quelques codes d'erreur à lui <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#dns-quic-error-codes> (RFC 9000, section 20.2), comme DOQ_INTERNAL_ERROR (quelque chose s'est mal passé), DOQ_PROTOCOL_ERROR (quelqu'un n'a pas lu le RFC, par exemple l'identificateur de requête était différent de zéro) ou DOQ_EXCESSIVE_LOAD (trop de travail tue le travail).

Un des gros avantages de QUIC est le 0-RTT, où des données sont envoyées dès le premier paquet, ce qui réduit nettement la latence <https://www.bortzmeyer.org/latence.html>. Cela implique que le client ait déjà contacté le serveur avant, mémorisant les informations qui seront données au serveur pour qu'il retrouve la session à reprendre. C'est cool, mais cela pose d'évidents problèmes de vie privée (ces informations mémorisées sont une sorte de "*cookie*", et permettent le suivi à la trace d'un client). D'ailleurs, en parlant de vie privée, le RFC signale aussi (section 5.5.4) que la possibilité de migrer une connexion QUIC d'une adresse IP à l'autre a également des conséquences pour la traçabilité <https://www.bortzmeyer.org/quic-tracking.html>.

Autre piège avec le 0-RTT, il ne protège pas contre le rejeu et il ne faut donc l'utiliser que pour des requêtes DNS idempotentes comme QUERY (le cas le plus courant) ou NOTIFY (qui change l'état du serveur, mais est idempotent). Bon, de toute façon, un serveur peut toujours être configuré pour ne pas accepter de 0-RTT, et répondre alors REFUSED (avec un code d'erreur étendu - RFC 8914 - "*Too Early*" <https://www.iana.org/assignments/dns-parameters/dns-parameters.xml#extended-dns-error-co

La section 5 du RFC liste les exigences auxquelles doivent se soumettre les mises en œuvre de DoQ. Le client doit authentifier le serveur (cf. RFC 7858 et RFC 8310, mais aussi RFC 8932, sur les bonnes pratiques). Comme tous les serveurs ne géreront pas DoQ, en tout cas dans un avenir prévisible, les clients doivent être préparés à ce que ça échoue et à réessayer, par exemple en DoT.

QUIC ne dissimule pas forcément la taille des messages échangés, et celle-ci peut être révélatrice. Malgré le chiffrement, la taille d'une requête ou surtout d'une réponse peut donner une idée sur le nom demandé. Le RFC impose donc l'utilisation du remplissage, soit par la méthode QUIC générique de la section 19.1 du RFC 9000, soit par la méthode spécifique au DNS du RFC 7830. La première méthode est en théorie meilleure car elle dissimule d'autres métadonnées, et qu'elle tient compte de davantage d'éléments, mais les bibliothèques QUIC n'exposent pas forcément dans leur API de moyen de contrôler ce remplissage. Le RFC recommande donc aux clients et serveurs DoQ de se préparer à faire le remplissage eux-mêmes (en relisant bien le RFC 8467 avant).

Un petit retour sur la protection de la vie privée en section 7. Cette section rappelle l'importance de lire et suivre le RFC 8932 si vous gérez un service de résolution DNS sécurisé (DoQ ou pas DoQ). Et elle insiste sur le fait que des fonctions de QUIC très pratiques pour diminuer la latence à l'établissement de la connexion, notamment le 0-RTT, ont des conséquences pour la vie privée, en permettant de suivre un même utilisateur (plus exactement une même machine). Et cela marche même si la machine a changé d'adresse IP entretemps. On peut aussi dire que le problème de QUIC est de permettre des sessions de très longue durée (que ce soit une vraie session unique, ou bien une « session virtuelle », formée en reprenant une session existante avec le 0-RTT) et que de telles sessions longues, excellentes pour les performances, le sont forcément moins pour l'intimité.

Les mises en œuvre de DoQ, maintenant. La société AdGuard <https://adguard.com/> a produit du code <https://github.com/AdguardTeam/DnsLibs>, dont un serveur et un client en Go, dnslookup <https://github.com/ameshkov/dnslookup>. (Regardez l'exposé du directeur technique de cette société à l'OARC <https://indico.dns-oarc.net/event/37/contributions/809/attachments/788/1381/doq_first_experience.pdf>.) Voici un exemple de compilation, puis d'utilisation de dnslookup :

```
% git clone https://github.com/ameshkov/dnslookup.git
%  cd dnslookup
% make
% ./dnslookup www.bortzmeyer.org quic://dns.adguard.com
```

On utilisait le serveur DoQ public <https://adguard.com/en/blog/dns-over-quic.html> d'AdGuard. Officiellement, NextDNS <https://nextdns.io/> en a également un mais je n'arrive pas à l'utiliser :

```
% ./dnslookup www.bortzmeyer.org quic://3e4935.dns.nextdns.io
...
2022/05/22 08:16:33 Cannot make the DNS request: opening quic session to quic://3e4935.dns.nextdns.io:853: t
```

Notez que si vous voulez utiliser dnslookup avec un serveur dont le certificat n'est pas valide, il faut mettre la variable d'environnement `VERIFY` à 0 (cela ne semble pas documenté).

Il existe une autre mise en œuvre de DoQ, quicdoq <https://github.com/private-octopus/quicdoq>, écrite en C. Elle dépend de la bibliothèque QUIC picoquic <https://github.com/private-octopus/picoquic> qui, à son tour, dépend de picotls <https://github.com/h2o/picotls>, dont la compilation n'est pas toujours évidente <https://github.com/h2o/picotls/issues/389>. D'une manière générale, les bibliothèques QUIC sont souvent récentes, parfois expérimentales, et ne marchent pas toujours du premier coup. Mais si vous arrivez à compiler les dépendances de quicdoq, vous pouvez lancer le serveur ainsi :

```
% ./quicdoq_app -p 8053 -d 9.9.9.9
```

Puis le client :

```
% ./quicdoq_app ::1 8053 foobar:SOA bv:NS www.bortzmeyer.org:AAAA
```

Le logiciel de test de performances Flamethrower <https://github.com/DNS-OARC/flamethrower>, écrit en C++ a une branche DoQ en cours de développement.

Le logiciel en Python aioquic <https://github.com/aiortc/aioquic> ne semble pouvoir interagir qu'avec lui-même. Vu les messages d'erreur (quelque chose à propos de "*not enough bytes*"), je soupçonne qu'il utilise l'encodage des débuts de DoQ, quand il n'y avait pas encore le champ de deux octets au début des messages. Même avec lui-même, il a des exigences pénibles en matière de certificat (pas de certificats auto-signés, obligation que le nom du serveur soit dans le SAN - "*Subject Alternative Name*", pas seulement dans le sujet).

Pour PowerDNS, il n'y a pour l'instant qu'un ticket <https://github.com/PowerDNS/pdns/issues/9897>. Et pour Unbound, c'est en cours <https://twitter.com/NLnetLabs/status/1529462240289726464>, ainsi que pour Knot <https://twitter.com/lundstromjerry/status/1529404041721262081>.

Dans les bibliothèques DNS, un travail est en cours <https://github.com/miekg/dns/pull/1377> pour go-dns <https://github.com/miekg/dns>.

Pour finir, vous pouvez regarder la présentation de DoQ par une des auteures du RFC au RIPE 84 <https://ripe84.ripe.net/archives/video/832/>. Et un des premiers articles de recherche sur DoQ est « "*One to Rule them All? A First Look at DNS over QUIC*" <https://arxiv.org/abs/2202.02987> ».